



DECISIVE ROUTING AND ADMISSION  
CONTROL ACCORDING TO  
QUALITY OF SERVICE CONSTRAINTS

THESIS

Cindy C. Reese, Captain, USAF

AFIT/GE/ENG/09-36

DEPARTMENT OF THE AIR FORCE  
AIR UNIVERSITY

**AIR FORCE INSTITUTE OF TECHNOLOGY**

Wright-Patterson Air Force Base, Ohio

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

The views expressed in this thesis are those of the author and do not reflect the official policy or position of the United States Air Force, Department of Defense, or the United States Government.

DECISIVE ROUTING AND ADMISSION  
CONTROL ACCORDING TO  
QUALITY OF SERVICE CONSTRAINTS

THESIS

Presented to the Faculty

Department of Electrical and Computer Engineering

Graduate School of Engineering and Management

Air Force Institute of Technology

Air University

Air Education and Training Command

In Partial Fulfillment of the Requirements for the  
Degree of Master of Science (Electrical Engineering)

Cindy C. Reese, BSCE

Captain, USAF


March 2009

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.


DECISIVE ROUTING AND ADMISSION  
CONTROL ACCORDING TO  
QUALITY OF SERVICE CONSTRAINTS

Cindy C. Reese, BSCE  
Captain, USAF


Approved:

  
Dr. Kenneth M. Hopkinson, PhD (Chairman)

18 MAR 09  
Date

  
Lt Colonel Stuart H. Kurkowski, PhD, USAF  
(Member)

18 Mar 09  
Date

  
Captain Ryan Thomas, PhD, USAF  
(Member)

18 MAR 09  
Date

## *Abstract*

This research effort examines, models, and proposes options to enhance command and control for decision makers when applied to the communications network. My goal is to research the viability of combining three students' past research efforts and expanding and enhancing those efforts. The area of this research is predicting a snapshot of the communications network, context-aware routing between network nodes, and Quality of Service-based routing optimization in order to create an intelligent routing protocol platform. It will consolidate efforts from an Intelligent Agent Based Framework to Maximize Information Utility by Captain John Pecarina, Dialable Cryptography for Wireless Networks by Major Marnita Eaddie, and Stochastic Estimation and Control of Queues within a Computer Network by Captain Nathan Stuckey. My research effort will create a framework that is greater than the sum of its individual parts. The framework will take predictions about the health of the network and will take the priority level of a commodity which needs to be routed, and then will utilize this information to intelligently route the commodity in such a way as to optimize the information flow of network traffic. Developing this framework will ensure that the forward commander and decision makers can make sound judgments at the right time using the most accurate information and on the proper communications network.

## *Acknowledgements*

First and foremost, I thank God and my savior Jesus Christ for this opportunity and I owe a large debt of gratitude to everybody who has helped me through this journey.

Cindy C. Reese

# *Table of Contents*

	Page
Abstract . . . . .	iv
Acknowledgements . . . . .	v
Table of Contents . . . . .	vi
List of Figures . . . . .	viii
List of Tables . . . . .	xi
List of Symbols . . . . .	xii
List of Abbreviations . . . . .	xiii
 I. Introduction . . . . .	 1
1.1 Motivation . . . . .	1
1.2 Quality of Service . . . . .	1
1.2.1 Differentiation of Traffic . . . . .	3
1.2.2 Admission Control . . . . .	4
1.2.3 Queuing . . . . .	5
1.2.4 Congestion Management . . . . .	6
1.3 Agent Based Framework . . . . .	6
1.4 Stochastic Estimation . . . . .	6
1.5 Dialable Cryptography . . . . .	7
1.6 Summary . . . . .	7
 II. Literature Review . . . . .	 8
2.1 A Technique for Adaptive Routing in Networks . . . . .	8
2.2 Dynamically Forecasting Network Performance Using the Network Weather Service . . . . .	11
2.3 Network Weather Service: A distributed Resource Performance Forecasting Service for Metacomputing . . . . .	13
2.4 The Network Tasking Order . . . . .	14
2.5 Stochastic Estimation and Control of Queues within a Computer network	16
2.6 Creating an Agent Based Framework to Maximize Information Utility	21
2.7 Dialable Cryptography for Wireless Networks . . . . .	28
 III. Decisive Routing and Admission Control Methodology . . . . .	 36
3.1 Overview . . . . .	36
3.2 Network Prediction Module . . . . .	37
3.3 Encryption Optimization Module . . . . .	46
3.4 Hybrid Agent Network Control Module . . . . .	51

	Page
IV. Decisive Routing and Admission Control Results . . . . .	56
4.1 Network Prediction Module . . . . .	56
4.1.1 Can Network Prediction Handle Noise? . . . . .	56
4.1.2 How Far in the Future are Predictions Precise? . . . . .	70
4.2 Decisive Routing and Admission Control Integration . . . . .	79
4.2.1 Experiment 1: Encryption Optimization . . . . .	79
4.2.2 Experiment 2: Preemptive Congestion Control . . . . .	83
4.2.3 Summary . . . . .	87
V. Conclusions . . . . .	90
5.1 Summary of Research . . . . .	90
5.2 Future Research . . . . .	91
Bibliography . . . . .	93
Vita . . . . .	95



## *List of Figures*

Figure		Page
1.1	Data Traveling Across Mixed Networks . . . . .	2
1.2	Internet Protocol Datagram Packet Header . . . . .	4
1.3	Basic Network Queuing Scheme . . . . .	5
2.1	M/M/k Queueing Scheme . . . . .	9
2.2	M/M/1 Queueing Scheme . . . . .	10
2.3	Network Weather Service (NWS) Structure . . . . .	12
2.4	Controlled System Configuration . . . . .	17
2.5	Linear-Quadratic-Gaussian . . . . .	18
2.6	Sensor Management and Scheduling example “In Harm’s Way” sce- nario Goal Table . . . . .	24
2.7	Sensor Management and Scheduling example “In Harm’s Way” sce- nario Goal Lattice . . . . .	26
2.8	Agent Based Framework Three Layer Architecture concept . . . . .	29
2.9	Conceptual dial settings for controller one scenario . . . . .	35
3.1	Network Prediction Module and Hybrid Agent Network Control Module . . . . .	39
3.2	Drop Tail Queue Representation . . . . .	40
3.3	Network Prediction Module and Hybrid Agent Network Control Module . . . . .	45
3.4	Encryption Optimization Module and Hybrid Agent Network Con- trol Module . . . . .	48
3.5	Sequencer Layer and Controller Layer . . . . .	53
4.1	Network Prediction Network Layout . . . . .	57
4.2	Compared Queue Size with Cholesky Factorization 100 . . . . .	58
4.3	Percent Difference between Actual Queue Size and Predicted Queue Size . . . . .	60
4.4	95% Confidence Interval Mean Difference in Queue Size . . . . .	61

Figure		Page
4.5	Compared Queue Size with Cholesky Factorization 50 . . . . .	62
4.6	Percent Difference between Actual Queue Size and Predicted Queue Size . . . . .	62
4.7	95% Confidence Interval Mean Difference in Queue Size . . . . .	64
4.8	Compared Queue Size with Cholesky Factorization 1 . . . . .	65
4.9	Percent Difference between Actual Queue Size and Predicted Queue Size . . . . .	66
4.10	95% Confidence Interval Mean Difference in Queue Size . . . . .	67
4.11	Interval Plot of Percent Differences with various noise parameters .	68
4.12	95% Confidence Interval Mean Difference in Queue Size . . . . .	69
4.13	Time Series Plot of fluctuations in Queue Size . . . . .	70
4.14	Time Series Plot of Actual Queue and 5 second Predicted Queue .	71
4.15	Percent Difference between Queue and 5 second Predicted Queue .	72
4.16	Histogram with normal distribution of queue sizes . . . . .	73
4.17	Total run of Actual vs Predicted Queue Size (5 seconds out) . . .	74
4.18	Time Series Plot of Actual Queue and 3 second Predicted Queue .	75
4.19	Percent Difference between the Actual Queue and 3 second Predicted Queue . . . . .	75
4.20	Histogram with normal distribution of queue sizes . . . . .	76
4.21	Total run of Actual vs Predicted Queue Size (3 seconds out) . . .	77
4.22	Time Series Plot of fluctuations in Queue Size . . . . .	78
4.23	Percent Difference between the Queue Size and 1 second Predicted Queue . . . . .	79
4.24	Histogram with normal distribution of queue sizes . . . . .	80
4.25	Total run of Actual vs Predicted Queue Size (3 seconds out) . . .	81
4.26	Total run of Actual vs Predicted Queue Size (3 seconds out) . . .	82
4.27	Percent Differences from Predictions at 1 sec, 3 sec, and 5 sec in future	83
4.28	Network Topology for Integration Testing . . . . .	85

Figure		Page
4.29	Network Topology for Integration Testing, Kalman Filter queue storing data . . . . .	86
4.30	Preemptive Congestion Control Code Snippet . . . . .	87
4.31	Preemptive Congestion Control Throttle Back Message First Occurrence . . . . .	88
4.32	Preemptive Congestion Control Throttle Back Message Second Occurrence . . . . .	88
4.33	Preemptive Congestion Control Response to Throttle Back Messages	89

## *List of Tables*

Table		Page
2.1	Sample Input File for Encryption Controller 1 [21] . . . . .	34
3.1	Sample Input File for Encryption Optimization Module [21] . . . .	47
3.2	Security and Performance Level Algorithms for Encryption Optimization Module [21] . . . . .	48
4.1	Encryption Optimization Input Parameters 1 . . . . .	84
4.2	Encryption Optimization Output 1 . . . . .	84
4.3	Encryption Optimization Input Parameters 2 . . . . .	84
4.4	Encryption Optimization Output 2 . . . . .	85

# *List of Symbols*

Symbol		Page
$\mu$	Queue Service Rate . . . . .	9
$\lambda$	Arrival Rate . . . . .	9
$\rho$	Calculated Arrival/Service time . . . . .	9
$\alpha$	Alpha . . . . .	20
$\beta$	Beta . . . . .	20
$\Gamma$	Gamma . . . . .	34
$\infty$	Infinity . . . . .	34
$\Phi$	Phi . . . . .	43
$\partial$	Partial Derivative . . . . .	43
$\Delta$	Delta . . . . .	43

## *List of Abbreviations*

Abbreviation		Page
IP	Internet Protocol . . . . .	1
QoS	Quality of Service . . . . .	2
DiffServ	Differentiated Service . . . . .	3
TOS	Type of Service . . . . .	3
HANC	Hybrid Agent for Network Control . . . . .	6
TCP	Transmission Control Protocol . . . . .	11
NWS	Network Weather Service . . . . .	12
NTO	Network Tasking Order . . . . .	14
GIG	Global Information Grid . . . . .	15
ATO	Air Tasking Order . . . . .	15
LQG	Linear-Quadratic-Gaussian . . . . .	18
ns2	Network Simulator version 2 . . . . .	19
AOC	Air Operation Center . . . . .	22
NIM	Network Information Maximization . . . . .	24
GMUGLE	George Mason University Goal Lattice Engine . . . . .	25
GnuPG	GNU Privacy Guard . . . . .	30
TCL	Tool Command Language . . . . .	38
glpk	GNU Linear Programming Kit . . . . .	47
BW	Bandwidth . . . . .	50
CPU	Central Processing Unit . . . . .	50
UBF	Unified Behavior Framework . . . . .	52
Mbs	Mega bits per second . . . . .	57
ms	millisecond . . . . .	57

# DECISIVE ROUTING AND ADMISSION CONTROL ACCORDING TO QUALITY OF SERVICE CONSTRAINTS

## I. Introduction

### 1.1 *Motivation*

Today information superiority is a key factor in military operations. According to Joint Pub 3-13, the ability to collect, process, and disseminate an uninterrupted flow of information while exploiting and/or denying an adversary's ability to do the same is the definition of information superiority [19]. It further states that to be successful, the conduct of operations requires access to information available outside the operational area and that warfighters need frequent, instant, and reliable access to information at locations in the continental United States as well as in theater. This research effort's aim is to provide a framework in which to give the most accurate information to the forward commander at the right time in such a way that they can achieve information superiority to accomplish the mission. This framework would concentrate on the area of the communications network. Throughout this thesis the terms data, information flow, traffic, and commodity are used synonymously to depict information that is placed on a communications network being sent from a sender and received at the proper destination.

### 1.2 *Quality of Service*

In a communications network, data or information is being transported from a sender to a receiver. This transportation of data may not occur in a direct route, the data may need to travel through several different routes to get to the receiver. The internet has become one reliant source to move this data from one place to another through the underlying protocol of the Internet Protocol (IP). This data or information may range from a simple document to complex video teleconferencing, and may

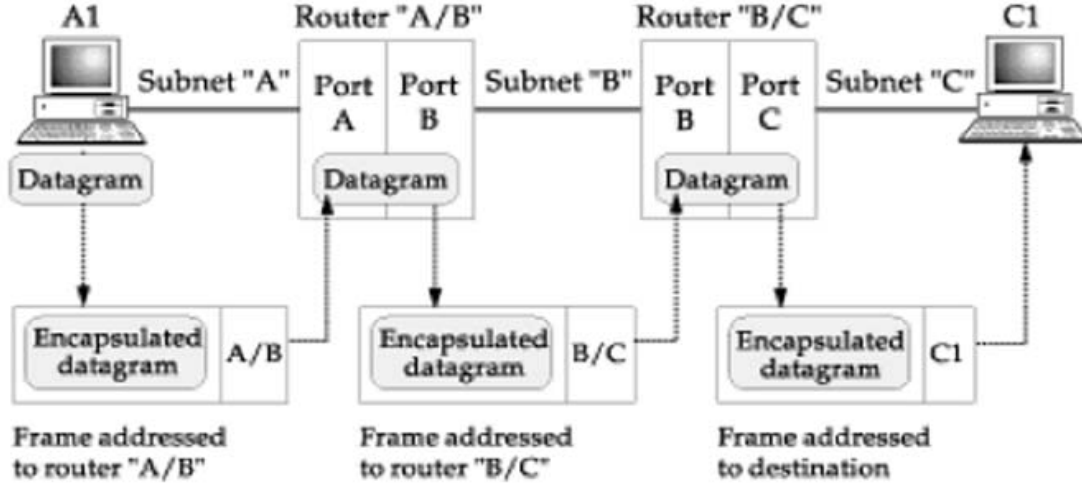


Figure 1.1: Data Traveling Across Mixed Networks

A datagram is created at source *A1* destined for *C1*. The datagram must be processed and sent through router *A/B*. Once there, the datagram must be processed again and sent through router *B/C*, since it is not at its intended destination. Then the datagram is determined to be destined for *C1* and routed to that location. At some point in this process, the datagram could be lost or dropped along its way [18].

have to travel across several subnetworks joined together by routers or relays. Since different rules could apply to different subnetworks and networks, IP, provides a universal way of packaging the data for transport across these networks' boundaries. IP provides this service with a best effort attempt to deliver the data, however, data may be dropped or lost along the way. The process of transport is depicted in Figure 1.1. Here you see that because of IP, data from source *A1* with a destination of *C1*, can travel across different networks to reach its intended destination. Each stop along the way, the data has to be processed and forwarded by the router; however, there aren't any guarantees that the data will make it to its destination. Because of other network traffic and congestion, the data could be lost and dropped along the way.

Hence an attribute that a customer would like to have and provider would like to give is the concept of Quality of Service (QoS). The concept of QoS gives types of standards in data or information delivery that rides on the internet backbone. Usually QoS describes what you get if you can guarantee the timely delivery of information on



networks, control bandwidth, set priorities for selected traffic, and provide a sufficient level of security [18].

Some ways to deliver increased QoS in networks are described in this research. A QoS enabled network must be able to handle different data being transferred from different subnetworks and networks like the one described above. This necessitates categorizing the data into types or classes and defining how each class or type is handled. All of the following aspects could be considered within the scope of QoS [4]:

- Differentiation of traffic (classification)
- Admission Control
- Queuing
- Congestion Management

The framework being proposed is a combination of research efforts in each of these area, creating a cohesive framework instead of individualized and multiple frameworks. The differentiation of traffic/classification will come under both Sections 2.7 and 3.3 for Dialable Cryptography as well as Sections 2.6 and 3.4 for Agent Based Framework. Admission Control, Congestion Management will fall under the section of Agent Based Framework. Finally, Queuing and Congestion Management will be discussed under Sections 2.5 and 3.2 for Stochastic Estimation.

*1.2.1 Differentiation of Traffic.* Differentiation of traffic pertains to being able to tell the difference between datagrams or packets of data. Normal operation is to treat every packet the same. If a packet needs to be handled differently, QoS looks at what would be the best method to handle the packet. One choice looks at the use of the packet header, Figure 1.2, however this would rely on more resources from the relay or router to look at this information and process it correctly. Another choice would be to just use the Differentiated Service (DiffServ) field or Type of Service (TOS), Figure 1.2 first row, starting at bit 8, this could be based upon several classification parameters [4]:

0	4	8	12	16	20	24	28	31
Version	Header Length	Type of Service	Packet Length					
Identification				D F	M F	Fragment Offset		
Time to Live		Protocol	Header Checksum					
Transmit IP Address								
Receive IP Address								
Options					Padding			

Figure 1.2: Internet Protocol Datagram Packet Header  
 Example of 32-bit datagram packet Header. Bit or placement is at top numbered 0 through 31. Placement of where information resides in the row and according to bit space.

- Service Mark
- Protocol
- Destination Protocol Port
- Source Protocol Port
- Destination Host Address
- Source Host Address
- Source Device Interface
- Any combination of above

*1.2.2 Admission Control.* This QoS attribute has to do with whether or not a datagram or packet is allowed through a router or relay. The router or relay can be configured to allow only certain types of traffic to be processed through. If the traffic doesn't meet this configuration, it is dropped or discarded. The parameters used in classification such as Service Mark, Protocol, etc. can also be utilized at this point

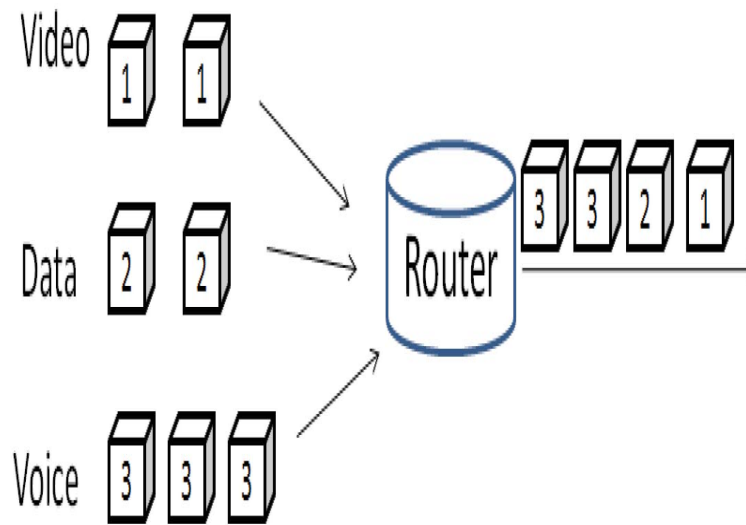


Figure 1.3: Basic Network Queuing Scheme

Basic depiction of how a queue process information. Video, Data, and Voice packets are going into a network router, when the router services the packet, it puts the packet on the outbound link in the order that it is being serviced.

to configure the router or relay to handle the classification types differently, thereby producing admission control of network traffic.

*1.2.3 Queuing.* This QoS attribute refers to the arrival of packets and what is done with the packet if the router is servicing other packets. Figure 1.3 shows a basic queue, there are video, data, and voice packets going through a particular router. Once the router services the packet, it is put on the outbound link of the router. If packets arrive faster than what can be serviced, the packet has to wait in line until it can be serviced by the queue. There are different queuing schemes available to manage this. Congestion and dropping packets occur at this level because the router is receiving packets faster than servicing and the queue or holding bin fills up. Once the queue is full packets begin to drop. The queueing scheme I'm running experiments with is drop tail, where the last packets/end packets drop while head/front packets are serviced.

*1.2.4 Congestion Management.* Lastly, Another QoS attribute is congestion management. This puts in place schemes that control congestion or when the queue backs up that will keep packets and data from dropping or being lost. During this research, a network predictor framework is being used to control some of the congestion in the network as well as an agent based framework that will react to congestion and controlling some of the traffic flow.

### ***1.3 Agent Based Framework***

One element of information superiority that is looked at is adaptability of the network in order to process information to meet the demands that are placed on the system. Hybrid Agent for Network Control (HANC) framework deals with the fluctuations of network bandwidth, due to anything from high traffic and low traffic demands, or failing network communications nodes. HANC provides a system that can make a decision on what to do with the piece of network traffic or commodity, whether to drop or to keep that commodity on target to its destination. It sends context aware messages to the neighboring nodes in the network.

HANC also considers the tradeoffs between bandwidth utilization and bandwidth maximization. It again brings the decision of considering the value and priority of the commodity and further goes into to a distributed tool for more than one decision maker where negotiation of competing mission objectives comes into play.

### ***1.4 Stochastic Estimation***

The Network predictor gives an alternative solution to the routing and congestion algorithm that are prevalently in use in the communications networks of today. These algorithms however, limit the ability of the network to be adaptable in a network environment that changes so frequently.

This research introduces the idea of using stochastic methods in determining the future state of the network by looking at some parameters that looks at the current

state. To accomplish this, the use of an Extended Kalman Filter was researched and modeled. The network is modeled as a feedback controller with stochastic controller.

### ***1.5 Dialable Cryptography***

The last research area that I'm combining is the ability to have adaptable cryptography that would be based upon the status of the network to ensure encrypted commodity is delivered. The research introduces the concept of adaptive security whereby it takes the collection of traditional security measures, vulnerability monitoring detection, and response and allows for manual or automated determination [21]

Because of a changing network missions and fluctuating bandwidth, the ability to be adaptive in cryptography prevents the decision maker from being pigeon-holed into one certain encryption scheme, it can be flexible to maximize information flow based on the priority of the commodity and network state.

### ***1.6 Summary***

The three research efforts that I am combining proposes options at solving the hindrances to network information maximization as well. The focus of an element of information superiority is superiority in cyberspace, to make sure at the core of the mission that there is optimal flow of information.

Since the advent of the internet, researchers have been attempting to improve Quality of Service in network communications. Being able to make dynamic decisions and dynamic action at the network nodes also increases QoS. The reason behind this has been to increase performance, decrease delays and to avoid congestion. I'm uniting three parts and creating a framework that is greater than the sum of its parts, proposing a possible solution to research in the area of computer networks.

## II. Literature Review

This chapter describes the research I am synthesizing to develop the framework for Decisive Routing and Admission Control According to Quality of Service Constraints. Details are given on the research theses I am bringing together as well as give examinations of some background material as how I'm building this framework.

### *2.1 A Technique for Adaptive Routing in Networks*

Some earlier research in the approach of adaptability in computer networks come from these authors Robert Boorstyn and Adam Livne [3]. They introduce adaptability in computer networks by taking a two-level adaptive routing approach [3]. This approach takes a view of a network node being a multiple server queueing system and then takes advantage of those qualities in obtaining some savings in average delay. An advantage of treating a node as a multiple server queue, according to the authors, provides a node with a processing factor approximately equal to the number of servers. A disadvantage however is that the control over good paths may get lost. This article deals with minimizing the disadvantages by introducing the two-level adaptive routing scheme.

The authors make the argument that time delays in a network would be reduced by the number of branches or edges a network node has, if operated as a queue of the number of those branches or edges. When time delays are reduced throughput can be increased. Good paths and faster performance is a key thought to this paper [3]. Further, if a commodity is not given a choice of output channels from the queue, it invariably might chose a bad path, a path that is congested.

The proposal of the two-level adaptive routing scheme offers each packet of data its own allowable channels which forces the use of the best path. The allowable branches are selected based upon the global network information topology, flows long term averages, and may be adaptive in a quasi-static way [3].

The network model was based upon a packet-switched communications network. Assumption of averaged length packets, input streams were independent and Poisson.

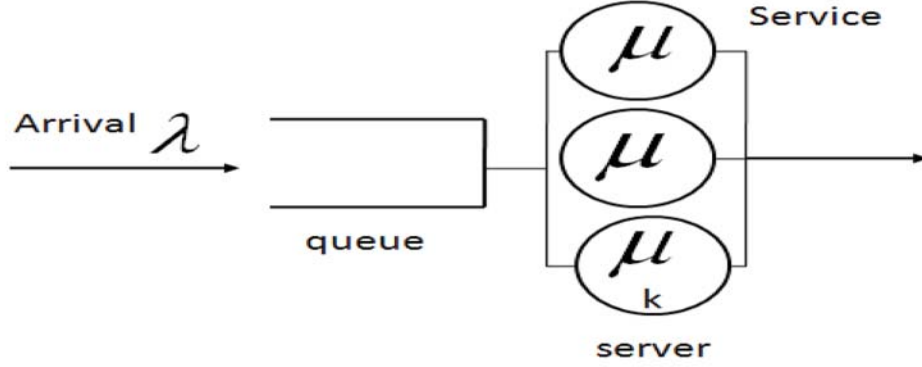


Figure 2.1: M/M/k Queueing Scheme

M/M/k Queueing Scheme is where one queue has multiple identical servers. The symbols  $\lambda$  represents the arrival rate of information and  $\mu$  represents the rate at which the queue can service or process that information. The arrivals are Poisson, and service time is exponentially distributed [15].

Using Kendall notation  $A/B/S$  to describe the queueing model,  $A$  represents the arrival process,  $B$  is the service time distribution and  $S$  as the number of servers [2]. Each node could be modeled via Figure 2.1 an  $M/M/k$  model vice Figure 2.2 an  $M/M/1$  queueing scheme model, where  $M$  is Markov property. If a node is set up as a multiple server, there seem to be an advantage according to the authors of at least  $k$ . According to the following equations [3]:

Using Queue Service Rate ( $\mu$ ) and Arrival Rate ( $\lambda$ )

M/M/k average waiting time

$$W_{(n)} = \frac{1}{\mu C} \frac{P_B}{k} \frac{1}{1 - \rho}$$

where  $\rho = \frac{\lambda_{(n)}}{\mu k C}$  and  $P_B$  is the probability that all servers are busy.

M/M/1 average waiting time at link 1

$$W_1 = \frac{1}{\mu C} \frac{1}{1 - \rho}$$

where

$$\rho = \frac{\lambda_l}{\mu C}$$

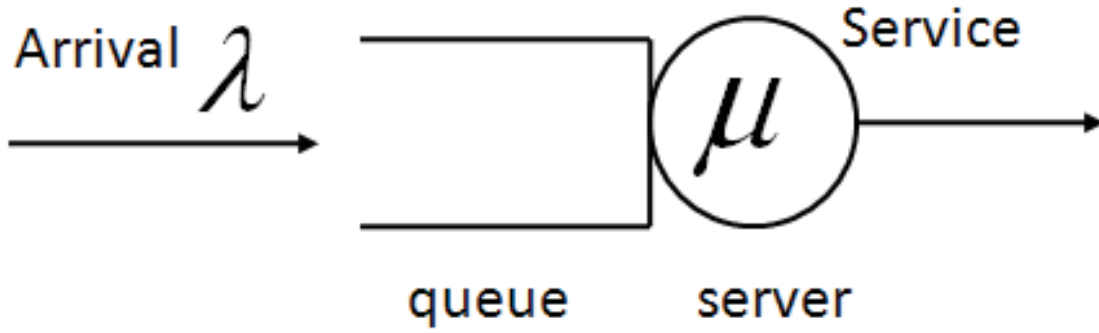


Figure 2.2: M/M/1 Queueing Scheme

[15] M/M/1 Queueing Scheme is the basic of the types of queuing. A single server queue model. The symbols  $\lambda$  represents the arrival rate of information and  $\mu$  represents the rate at which the queue can service or process that information. The arrivals are Poisson, and service time is exponentially distributed.

The first level was based on the global status of the communications network and using that information to make decisions. This is also the idea I'm using for the Kalman Filter Stochastic Controller. These decisions are somewhat static however, in that the routing paths are being laid out beforehand. These routing paths hardly vary except in cases of where extreme congestion and link failure occur.

The routing topology is determined between the communicating network nodes. The authors determine the paths not based upon the best route but rather on the possible route with the least number of hops and little congestion, on parameters that seemingly makes the route a good path. Due to the large possible paths that are generated, the premise is to allow the commodity to use any outgoing link that the node may have, thereby making a tradeoff with taking advantage of modeling a node as a multiple server queue versus traffic load. By the authors not forcing a choice when there are similar paths, they state that there is a  $k$ -fold improvement in delay [3], where the  $k$  is the number of outgoing links.

The second level of the routing is implementing queuing disciplines at each node. This discipline entails a decision of what output channel to use based upon whether or not a commodity was already waiting on the outbound channel.



Strategies that the authors settled upon after modeling a multiple server queue was to give priority to a dedicated queue and when the dedicated queue was empty then service the non-dedicated queue and secondly when packets arrive to a queue then join the shortest queue, this then would produce a behavior similar to a  $M/M/k$  queue.

The results of this simulated model showed that if 15-20% of the traffic is not dedicated and the choice of traffic is not apportioned to disjoint sets of servers, then the node behavior will achieve more than half the improvement of an  $M/M/k$  queue as well as if the traffic having the choice is too light or if the servers are in isolated groups, then the advantage is reduced [3].

The research of the Decisive Routing According to Admission Control and Quality of Service Constraints is similar in that there is a choice in a decisive decision based upon either the global foresight picture of the network status or based upon the current congestion conditions at the node using the Network Prediction Module and the HANC. It will provide a proactive approach or reactive approach to the handling of routing network traffic.

## ***2.2 Dynamically Forecasting Network Performance Using the Network Weather Service***

The article of “Dynamically Forecasting Network Performance Using the Network Weather Service” [23] is in the direction of my research. Its focus is on predicting Transmission Control Protocol/IP (TCP) end-to-end throughput and latency that is attainable by an application using system located at different sites [23]. It looks at these different sites and labels as metacomputer, meaning distributed computers whereby they are interconnected but separated and used as high-performance computational platforms [23].

A pictorial representation of the distributed service is shown in Figure 2.3. In this figure, there are distributed computer systems with a CPU and Memory sensor

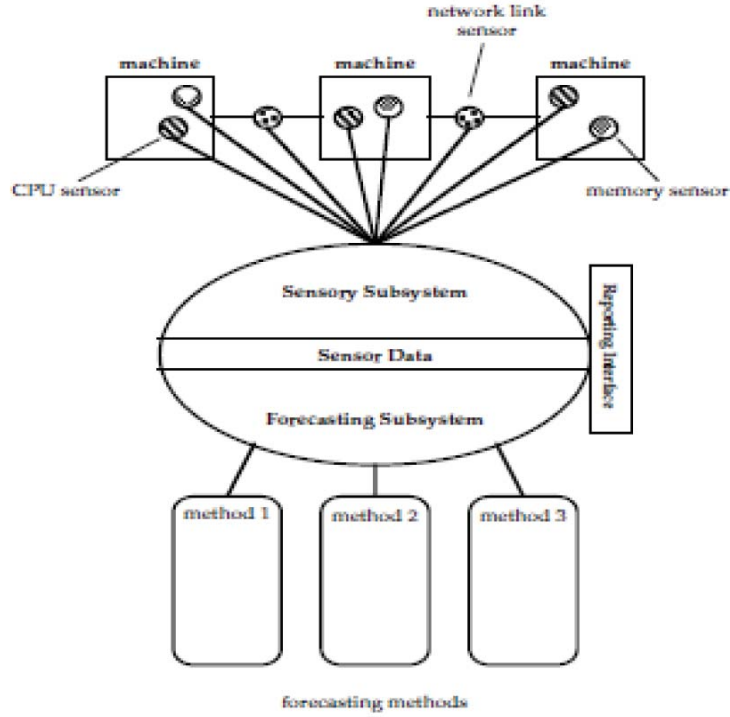


Figure 2.3: Network Weather Service (NWS) Structure

Sensory data is compiled into a logically central database, each sensor takes performance measurements periodically and time stamp it. The resulting collection of measurements for a time series describing the behavior of the resources. The NWS forecasting system then generates predictions of what the performance will be for the given resource [23].

mechanism that reads and gathers the data regarding the conditions of the system. Then uses this data to perform forecasting methods and calculations to predict the future service of a particular resources in the system. This Network Weather Service (NWS) model uses numerical models to generate forecasts of what the conditions will be for a given time frame [23]. The NWS tool has three characteristics that it performs, sense and gather data of the performance of a particular resource that is on the network, forecast the future performance of these resources based upon the sensory data gathered, and then disseminates and relays this information for appropriate decision making. The numerical models used to determine the resources performance at a given time frame are broken down into three categories, mean-based, median-based, and autoregressive methods of prediction determination. However different

from the Extended Kalman filter prediction model that is being implemented, this model has six predictive methods that at the onset of the implementation it maintains all six calculations for each resource then it uses an error measure to produce an overall metric.

### ***2.3 Network Weather Service: A distributed Resource Performance Forecasting Service for Metacomputing***

Network Weather Service: A distributed Resource Performance Forecasting Service for Metacomputing is a follow on to the above Section 2.2. This however looks at resources at the application level and not at the network transport level.

The paper describes how to provide accurate forecasts of dynamically changing performance characteristics from a distributed set of metacomputing resources [24]. This paper relates to the QoS research in forecasting and gathering useful data from dynamically changing computer networks. The research takes an approach from distributed resources and being able to gather information to look at what the load of the resource will be and predict what the performance of that resource would produce. An example would be a customer on a college campus wants to use a printer to print out their thesis of 200 pages, they would like to know what is the best time to use that resource, what time would give them the maximum benefit. There are similarities to my research in that it looks at some parameters of the system to determine the load of the queue in a network to make a decision on what is the best benefit for the sending the data across the network. The network weather service is a distributed, generalized system for producing short-term performance forecasts based on historical performance measurements [24]. This service occurs at the application level. The architecture of the NWS was based upon meeting the four characteristics. Predictive accuracy in the estimations, non-intrusiveness in the added load that the NWS might add, execution longevity, and ubiquity across the system.

The components of the NWS is a persistent state process that gets and saves the stored measurements, a name server process to bind the client and server requests, a sensor process that takes the performance measurements from a specified resource, and a forecaster that produces the predictions.

The performance monitoring takes the parameters of CPU availability, TCP connection time, latency, and bandwidth for the end-to-end connections. Some of the Kalman filter takes as inputs the latency and bandwidth.

Network sensors are in place to take the measurements and feed it to the forecasters of the system. The network sensors use an active probing to the network connections of interest. The sensor measures small-message round-trip time, large message through-put and TCP socket connect-disconnect times.

Forecasting predictions are calculated by taking the stored time-stamped performance measures that are put in a circular queue type fashion and generates future measurement values. Every forecasting model produces predictions for each measurements and a cumulative error, the model then determines the best prediction to use from the model with the lowest prediction error [24].

To alleviate network congestion and overload of messages being sent for forecasting and I'm alive messages, network sensors are organized as sets of cliques. Only a single clique member will conduct probes at any one time using a token passing algorithm. In the Hybrid Agent Based framework, congestion is handled by throttling back on the number of packets processed by sending raise or lower threshold messages.

## ***2.4 The Network Tasking Order***

This section introduces the idea of having a Network Tasking Order (NTO) [5] which would give the Decisive Routing and Admission Control According to Quality of Service Constraints framework the input necessary to determine outrange predictions of the global network.

The idea behind the NTO is to give the commander or decision maker the tools needed to manage the network usage and gives visibility into the load of the network at a certain time period based upon the mission task requests.

The author of the article, The Network Tasking Order, motivation is behind the fact that the Global Information Grid (GIG) is static rather than dynamic and incapable of supporting net-centric warfare and net-centric operations [5] and proposing the idea of having a network tasking order energizes the capability of the GIG of responding rapidly to the mission needs. The aim as the author state is to move information where it is needed, whether localized from soldier to soldier on the battlefield or worldwide from soldier to the decision makers [5].

The NTO would be similar to the format and layout of an Air Tasking Order (ATO) and contain information that will reflect the missions and request of network resources. The ATO is used as a daily schedule for air missions. The ATO is broken down into the geographical location of the mission, the type of aircraft needed to carry out the mission, and the details on the communications between what commands will utilize. Similarly, the NTO can be developed with the characteristics of the ATO with added communications network details.

The NTO can be integrated in the prediction level of the Decisive Routing and Admission Control According to Quality of Service Constraints framework, where it can provide the basis for making decisive decisions for dynamic network. It can be integrated in the Hybrid Agent Network Control level in that it can be used to prioritize the network missions so that the nodes can reactive smartly and dynamically regarding the decisions to make regarding the commodity that the node is processing. Further, the NTO can be integrated in the encryption level of the framework as well. This integration can be developed once the commodities and encryption algorithms are calculated to produce a list of the desirable ways in which to send an encrypted commodity in order to optimize the network.

A NTO makes sense in that a commander can see a global picture of the available resources in which to transport the needed data to the target destination. A commander can be given a choice if there is upcoming congestion on the network or predicted congestion at a certain time frame, which would give them time to adjust what is the best method or route in which to transport the information. The scenario given in the paper was the choice of using an available aircraft in which to ferry the data to the destination, thereby avoiding congestion and delays of a link that is already saturated with data. The network prediction module and hybrid agent network control modules can be used in these types of circumstances, giving a global picture, and providing some sense of congestion control at node level give a commander the tools needed to make decisive decisions on transporting data that is critical.

## ***2.5 Stochastic Estimation and Control of Queues within a Computer network***

The research [20] proposed the development of a better process to deliver commodities across a communications network using information about the queue at the node. The author points out the changing and dynamic nature of the network and the need to have a routing and congestion control algorithm that suits the changes to provide optimal routing and congestion control. To accomplish optimal routing and congestion control, this work establishes the use of available data about the network state and produce some predictions of the future network state.

Creating a network state model is taken from the network tomography field, where inferential network monitoring is introduced. This monitoring is taking measurements within a network that doesn't have to involve internal network devices that might prove to be very difficult to obtain. These measurements are further used in the calculations in deriving the state of the network.

By using a feedback controller as in Figure 2.4 where the feedback is the measurements from a dynamic system such that it takes input from disturbances, corrup-

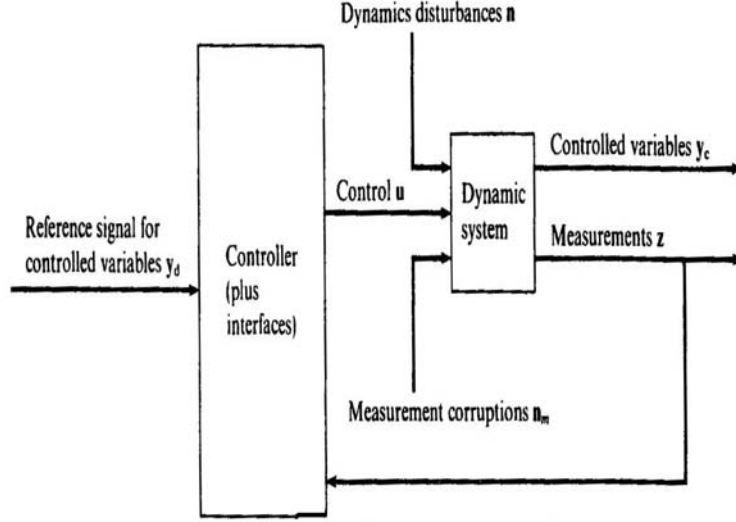


Figure 2.4: Controlled System Configuration

A Controlled System Configuration whereby the *Control  $u$*  delivers to the Dynamic System such that the *Controlled variables  $y_c$*  match the *Reference Signal  $y_d$*  as close as possible. *Dynamic disturbances  $n$*  also effect the Dynamic System undesirably. In order to observe these disturbances, *Measurements  $z$*  of the Dynamic System are taken and are fed back into the controller. The *Control  $u$*  is computed based on the feedback from the measurements which also included some type of *Measurement corruptions  $n_m$*  [20].

tion measurements, and input from the controller, you can develop a network control algorithm. The feedback controller is liken to the Heating Ventilation and Air Conditioning system of a building, where one can change the temperature of the system based upon the current temperature. A stochastic controller is developed from this concept, this controller would use a Kalman filter to estimate the state of the system by using measurements and previous control inputs.

Based upon a control system configuration, a computer network would be the dynamic system. To get the desired service from the computer network, the controller would give the computer network a control parameter based upon the manipulation of the controlled variables to match the reference signal as much as possible. The controlled variables consist of the performance metrics of interest such as network delay, data throughput, and congestion levels further, the control inputs in to the

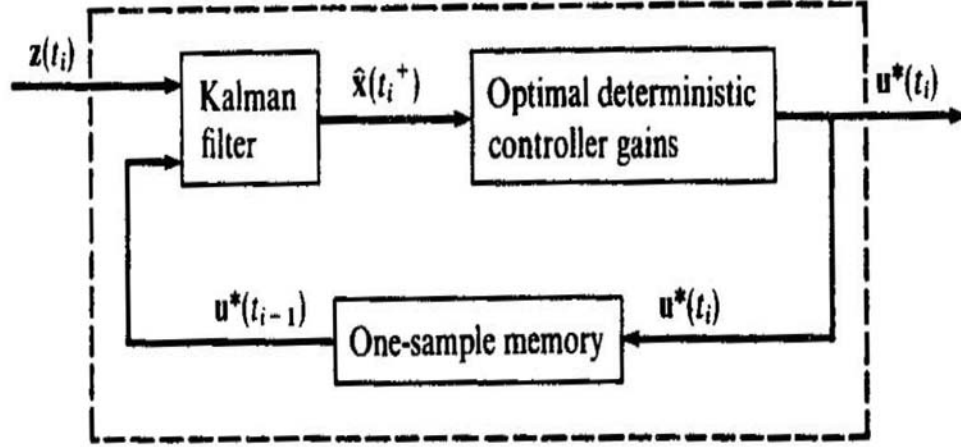


Figure 2.5: Linear-Quadratic-Gaussian

The stochastic controller uses a Kalman filter to estimate the state of the system  $\hat{x}$  based on the measurements and previous control inputs. Optimal control inputs  $u^*$  are computed based on these state estimates and the optimal controller gains [20].

dynamic system, could be computed sending rates, routing table entries, optimal data packet sizes, for example [20].

Again based upon the control system configuration, the controller was developed by applying stochastic control theory using a Kalman filter. The controller was developed based upon the stochastic Linear-Quadratic-Gaussian (LQG). Figure 2.5 depicts the design. The controller takes unknown quantities, dynamics disturbances and measurement corruptions to be used as parameters to evaluate for control purposes and feeds back into the dynamic system. The Kalman filter is used to estimate the size of network queues and total packet arrival rates.

The Kalman Filter application involves using a system that models as a set of differential equations in the continuous-time case or as a set of difference equations in the case of a discrete-time system. The model is used to propagate the estimate of the state of the system forward in time until a measurement is received. Then the system state attained from the measurement is compared to the estimate of the system state and combined in an optimal manner [20]. The Kalman filter measurements also takes



into account the uncertainty factor and error, with the measurement, a calculated weight is given to the system state estimate.

The network queue controller model was developed based upon the assumption of having Markov properties and LQG. The Markovian properties presents itself as having characteristics of not depending on the past state of the system. The Markovian definition states that the conditional probability distribution of future states of the process, given the present state and all past states, depends only upon the present state and not on any past states. The equations were derived from the premise of the transient behavior of the network queues using Marcum's Q-Functions.

The author derives the equation from the Markov property in the probability of going from one state to another in the time starting at some point and ending at some point in the future. These equations describe the behavior of the queue. Since it's referring to the rate of change it's a derivative type equation.

Even though a queue has a finite space in which to house packets waiting to be serviced, the equations give a model bases on how the queue will behave given the packets it receives and because the queue size is measured by what is in the queue at any particular time slice not the maximum size that a queue may hold. The derived equations indicates how to calculate the size given that the queue fluctuates in any given time and are taken from [20]. The expected value of the queue size is calculated from Equation 2.1, the expected value of the number of packets in the queue as a function of time 2.2, and 2.3 which provides the basis for the system model used by the network state estimator and network queue controller. These equations are done in `Matlab`<sup>®</sup> [9] and called from an Network Simulator Version 2 (ns2) [13] method.

$$\begin{aligned}
p_n(t) &= (1 - \rho)\rho^n \\
&+ \rho^n(\rho Q_{n+n_0+2}(\alpha, \beta) - Q_{n+n_0+1}(\alpha, \beta)) \\
&+ \begin{cases} Q_{n-n_0+1}(\beta, \alpha) - Q_{n-n_0}(\beta, \alpha) & n > n_0 \\ Q_1(\alpha, \beta) + Q_1(\beta, \alpha) - 1 & n = n_0 \\ Q_{n_0-n+1}(\alpha, \beta) - Q_{n_0-n}(\alpha, \beta) & n < n_0 \end{cases} \quad (2.1)
\end{aligned}$$

where

$$\rho = \frac{\lambda}{\mu}, \alpha = \sqrt{2\rho\mu t}, \beta = \sqrt{2\mu t}$$

and  $Q$  is given by

$$Q_m(\alpha, \beta) = \exp\left(-\frac{\alpha^2 + \beta^2}{2}\right) \sum_{k=1-m}^{\infty} \left(\frac{\alpha}{\beta}\right)^k I_k(\alpha, \beta)$$

For  $n_0 \geq 1$

$$\begin{aligned}
E[n(t)|n_0] &= \frac{\rho}{1-\rho}(1 - Q_{n_0+2}(\alpha, \beta)) \\
&- \frac{1}{\rho^{n_0}(1-\rho)}(1 - Q_{n_0+2}(\beta, \alpha)) + n_0 Q_{n_0+1}(\alpha, \beta) \\
&+ \rho\mu t Q_{n_0+2}(\alpha, \beta) - \mu t Q_{n_0}(\alpha, \beta) \quad (2.2)
\end{aligned}$$

For  $n_0 = 0$

$$\begin{aligned}
E[n(t)|n_0 = 0] &= \frac{\rho}{1-\rho}(1 - Q_2(\alpha, \beta)) \\
&- \frac{1}{1-\rho}(1 - Q_2(\beta, \alpha)) \\
&+ \rho\mu t Q_1(\alpha, \beta) - \mu t(1 - Q_2(\beta, \alpha)) \quad (2.3)
\end{aligned}$$

The Kalman filter controller is a discrete-discrete extended Kalman filter. This was chosen because the communications network is nonlinear and discrete and the measurements are discrete. The derived equations can be seen in [20]. The steps for the filter is the time propagation and the measurement update. The following equations are computed in `Matlab`<sup>®</sup> [9] and called from an ns2 [13] method.

A discrete-time nonlinear stochastic controller is further designed to regulate the network queue using the basis that the network is a discrete-time and nonlinear. This controller is what regulates the flow control. The author looked at using dynamic programming for an optimal solution, however because of it's complexity, he chose to used a linear perturbation control law and steady state constant gain control law. The derived equations are available in [20].

Finally, the network state estimator and the network queue controller is developed. The network state estimator will estimate the state of the network based on measurement of the queue size.

## ***2.6 Creating an Agent Based Framework to Maximize Information Utility***

Another aspect of the framework that is being developed, is the ability to react and make decisions as to what to do with the commodity at the node when congestion is imminent. This research effort looks at such possibilities in the direction of cyberspace and Air Force mission accomplishment. Since technology is moving into the cyberspace era, the Air Force is headed in that direction as well. The Secretary of the Air Force states "A great deal of our combat capability operates in cyberspace: command and control systems as well as the intelligence, surveillance, and reconnaissance platforms that ensure battlefield awareness" [14]. Further, this concept has been adopted into the Air Force Doctrine and Mission. Thus, the mission of the United States Air Force is to deliver sovereign options for the defense of the United States of America and its global interest to fly and fight in Air, Space, and Cyberspace [17].

Pecarina took the approach in his thesis [14] to create a framework that solved at least five requirements of an optimal network system. His point of view is from an Air Operations Center (AOC). The first requirement is to create a system that makes “thoughtful” decisions on what traffic to drop and what traffic to keep in relation to mission importance [14]. The requirement is to cover the aspect of traffic being classified into a low, medium, and high bandwidth request. Low bandwidth demand traffic is passing sensor data, medium bandwidth demand traffic is passing file and message traffic, finally high bandwidth traffic is passing very large data files and multimedia feeds. If the network gets inundated with large amounts of traffic with no real decision making, very important traffic would be lost and the system could become congested and slow down the arrival of that very important message or multimedia.

The second requirement looks at the ability of the network to strike a balance between the available network resources and meeting the current mission objectives. This requirement refers to looking at the trade off between important information and bandwidth utilization [14]. The scenario given is that of using the communications link for a critical video stream which takes up a large portion of the available bandwidth, however there are lower priority items that only use a small portion of the bandwidth, therefore there is a choice forcing “poor utilization” in order to satisfy mission objectives.

The third requirement is to create a system that makes distributed decisions about what information needs to flow from source to sink [14]. This requirement focus on the fact that there may be several routes from the source node to the destination that network traffic could take and the volume of traffic that need to be transported. If network traffic has many choices in which to use, and the volume of traffic may be heavy on only certain routes, then some smart decisions are needed to ensure that the proper and critical information is received at its designated destination via some dynamic decisions about which route to take. This also brings to mind the earlier discussed section on Adaptive Routing in Networks, using the fact that if a node has

several outgoing paths or links, let the path be chosen dynamically based upon current traffic congestion. If there aren't any way to make decisions dynamically, as Pecarina points out that if the traffic has a static route, it must compete for the common bandwidth resources of the entire network, which could then result in dropped, lost, or delayed traffic.

The fourth requirement is to create a system that must adapt to periods of bandwidth fluctuations [14]. This requirement looks at the possibility of a network that has a number of different assets using the network at different times with a variable rate of usage. This also coincides with the earlier discussed section on the Network Weather Service, looking at available resources and the performance. The scenario given was the idea of satellites and airplanes use the network for limited but predictable times and the intermittent connections of wireless links [14]. Having an option to decide what is the best practice to use is what the Decisive Routing and Admission Control according to Quality of Service Constraints framework is trying to achieve.

The fifth and final requirement is to create a system that negotiates multiple, competing mission objectives [14]. The idea behind this to address that the system has multiple users, not just the AOC commander and multiple mission objectives. If there is a standard in which mission objectives are used, then classification schemes could be standardized. If the framework is then deployed at any number of nodes, then the same classifications will be decided upon and handled the same, what is critical will remain critical across the communications network channels.

The above mentioned requirements are also key and relevant to the Decisive Routing and Admission Control According to Quality of Service Constraints framework. The framework is acting as the glue that is bringing together different and distinct frameworks but meeting the same overall requirements that will enhance the overall way decisions are being made at the relay and routing level. The framework I am proposing will continue to use these requirements as a baseline of characteristics.

Goal Number	Goal	Included Goals
1	to obtain and maintain air superiority	2, 3, 4, 5
2	to minimize losses	6, 7, 8
3	to minimize personnel losses	6, 7, 8
4	to minimize weapons expenditure	6, 8
5	to seize the element of surprise	8
6	to avoid own detection	9, 10
7	to minimize fuel usage	10, 11
8	to minimize the uncertainty about the environment	12, 13
9	to navigate	15, 16
10	to avoid threats	15, 16
11	to route plan	15, 17
12	to maintain currency of the enemy order of battle	14, 16
13	to assess state of the enemy's readiness	14
14	to collect intelligence	15, 16, 17
15	to track all detected targets	
16	to identify targets	
17	to search for enemy targets	

Figure 2.6: Sensor Management and Scheduling example “In Harm’s Way” scenario Goal Table  
The figure displays a list of mission goals taken from a compilation of USAF doctrine manuals for a “In Harm’s Way” sensor scenario. The first column is the node number assigned to the goal, next column is the stated goal, and the last column indicates the goals that are included in attaining that particular goal. The bottom three goals are observation functions to track, identify, and search [10].

In this reading [14], I also picked up the naming convention called Network Information Maximization (NIM), NIM is the ability of a network of queues, schedulers, and routers to produce the most useful data to a user or users given time and bandwidth constraints [14]. I’ve also adopted this concept because the framework is to maximize the information that can be delivered across the network. Further definitions used within Pecarina’s work that I also chose to use is the meaning of information flow and the value that is placed on the commodity. Information flows are more than just file or message transfers, long standing, heterogeneous, and sometimes discontinuous blocks of data answering specific information requirements for a user according to [14].

The decision to allocate resources has to begin with the classification of the commodity, as mentioned in the introduction, there are several areas data can be differentiated by. In [14], the author makes mention that the classification of this commodity can be broken down by sources of mission association, timeliness and customer input which gives a quantitative descriptor that can be used in the decision

making process. This quantitative description is used to prevent overloaded information, missed information, or the difficulty in pinning down the intent if using vague and non-descriptive qualitative descriptions such as low, fair, or high value. The end result of the classification gives the commodity a utility value which the network can use to form a intelligent decision. The author ties mission objectives and goals to information utility values.

A software system called George Mason University Goal Lattice Engine (GMU-GLE<sup>®</sup>) is a resource that could be used to break down these goals and objectives into weighted values based upon the numbered goals. The goals are in a listed and numbered order based from the highest mission objectives first. A list of possible goals and weights is depicted in Figure 2.6 taken from a sensor management and scheduling example.

Figure 2.6 displays a list of mission goals taken from a compilation of USAF doctrine manuals for a “In Harm’s Way” sensor scenario. The first column is the node number assigned to the goal, next column is the stated goal, and the last column indicates the goals that are included in attaining that particular goal. The bottom three goals are observation functions to track, identify, and search [10]. Goal lattices are a method for ordering the goals of a system and associating with each goal the value of performing that goal in terms of how much it contributes to the accomplishment of the topmost goal of a system [8]. This can be related to the information flows/commodity whereby as the commodity enter the network, it can be given a weight based upon the already determined goals and objectives of the mission. Once the goals are categorized and prioritized, a lattice can be determined as in Figure 2.7. The figure exhibits a lattice based upon the goals of Figure 2.6. The decimal number in the figure are the weight associated with the goal. The top level node is goal number 1, the next level is all goals included in number one and a line is drawn. In this example is goals 2, 3, 4, and 5. Each subsequent level down follows this pattern, drawing a line to represent the relationship to the higher goal. This concept can relate to the communication network in that, as missions or commodities are to be put on

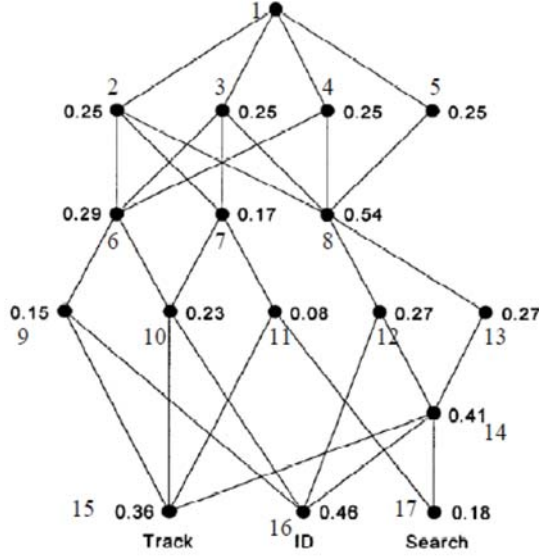


Figure 2.7: Sensor Management and Scheduling example “In Harm’s Way” scenario Goal Lattice. The decimal number in the figure are the weight associated with the goal. The top level node is goal number 1, the next level is all goals included in number one and a line is drawn. In this example is goals 2, 3, 4, and 5. Each subsequent level down follows this pattern, drawing a line to represent the relationship to the higher goal [10].

the network link, it would be racked and stacked against the goal and goal number of the lattice and given a weight to reflect such decision. This weight will be given based upon the mission and objective of the customer or commander. This process can give the commander a better idea of where it would scale in the bigger picture of the network routing process. Then later on discussed, encryption module process can be combined at this level to further the reasoning behind goal decomposition to get optimal results in the network taskings. More detailed information pertaining to this encryption scheme and algorithm is discussed in the section Dialable Cryptography. When an information flow arrives is a crucial aspect as well. Lastly customer input could be used to classify information, Mission Critical, Mission Essential, and Mission Non-essential. Pecarina states that proper classification supports the ability to index information flows for scheduling and the allocation of resources and therefore used as an information utility during the scheduling process.

This classification can be broken out on the NTO. The order can give the type of taskings as well as the mission and goal objectives weighted value based upon a stan-



dardized set of values. This document is what can begin some preliminary decisions about the way network commodities are handled at each node. Again encouraging the dynamic ability of the nodes.

The next step in Network Information Maximization that Pecarina contemplated was taking the classified commodities and producing an optimal schedule to be processes across the network [14]. This step looks at meeting the requirement to create a system that makes distributed decisions about what information needs to flow from sender to destination. The author solves this by using the approach of the restless bandit problem, using an adaptive greedy algorithm that will take a set of commodities and index and sort based upon the classification of its utility. And again utility corresponds mission objectives or goal of the commodity to be transported. Once this set of optimal scheduled commodities is complete, bandwidth has to be allocated in order to move the commodity across the network. Even further, the concept of the Network Tasking Order can be used for scheduling and bandwidth allocations, with the addition of the network predictor module, bandwidth can be allocated based upon future events.

Allocation of resources is a crucial step in network information maximization. Pecarina points out with the use of priority queues at the network nodes, even though the lower utility flows, less important commodities, will be dropped and the higher utility flows, commodities of greatest importance, will get through to the receiving customer, the second highest is not guaranteed that it will be delivered. Thereby producing results that are not desired. Secondly, if resource allocation is too narrow in its scope of parameters, the utilization of the resource may not be considered. Utilization of the resource may be taken up by a commodity that in not of a greater priority, while a commodity that is of greater importance cannot use the link.

Therefore, an agent based architecture was presented via the thesis to solve the problem of bandwidth allocations system wide as to minimize congestion and bottlenecks for the flow utility. An agent is an entity that uses machine logic to

analyze state and perception to act on its environment [14]. The concept of using robot control concepts in the communications network was introduced in the thesis, *Creating an Agent Based Framework to Maximize Information Utility*. It takes the three layer architecture, see Figure 2.8, you would give the deliberator layer the input, for example from network owner or Network Tasking Order. The sequencer layer takes feedback from the deliberator and builds the library of the wanted behavior or actions wanted to be performed. The controller layer then controls the execution of that behavior to accomplish the task of the robot. The sensors take in and gather data from the environment and saved in the state module eventually, the process repeats itself based upon the data gathered. An agent deployed in every node that considers the state of the network and selects actions that seek to maximize the aggregate utility of the network flows, while balancing the efficient usage of resources [14] can therefore be achieved based upon this type of model. Since a communications network is a distributed system, the controller mechanism has to accommodate this type of environment, therefore, a multi-agent system is utilized. This would be a similar architecture as the Network Weather Service where there are sensors at the corresponding resource nodes.

## ***2.7 Dialable Cryptography for Wireless Networks***

The *Dialable Cryptography for Wireless Networks* [21] thesis work covers another part of the Decisive Routing and Admission Control According to Quality of Service Constraints framework. The objective of this thesis was to develop an adaptive cryptographic protocol which allows users to select an optimal cryptographic strength and algorithm based upon the hardware and bandwidth available. Sensitive or classified information can be transferred wirelessly across unsecured channels by using cryptographic algorithms [21].

This framework was chosen because it can be a useful tool with added benefit in the adaptivity of a network. This tool proposes the creation an optimal schedule and best course of action in the decision process at each agent node when there are

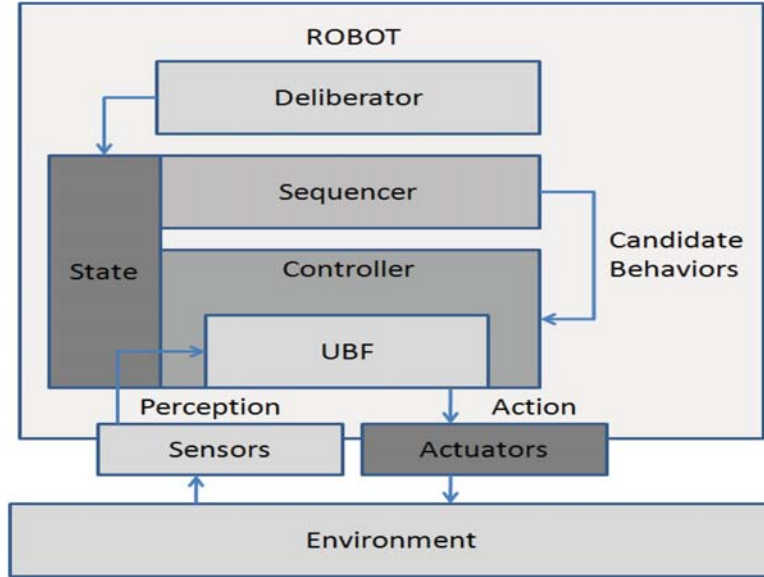


Figure 2.8: Agent Based Framework Three Layer Architecture concept

Deliberator layer acts as the input of mission objectives and goals. The sequencer layer takes in data from the controller and feedback from the deliberator to build a library of behavior sets and actions. The controller layer controls the execution of behaviors selected to accomplish mission objectives. Sensors take in data and store in state component where the controller then reacts to the current state of the system [14].

encryption and decryption requirements. I believe this is crucial as Eaddie states cryptography continues to play a major role in the military and in the public sector. Static security methods and statically chosen cryptographic schemes cannot adjust to changing environmental factors especially in the wireless environment [21]. Therefore, there is a need for decisive decisions in what is the best security measure for the environment at hand and carefully chose an optimal method of delivery.

The idea of dynamically changing the security of a system is important in wireless ad hoc and sensor networks where critical resources such as battery life, memory, computational power, and bandwidth are not constant nor necessarily predictable [21]. The research behind having an adaptive security posture will help provide solutions for the military requirements in the area of network-centric warfare and secure teleconferencing, telephony and imagery. Decisive decisions can be made possible by using a framework that can take the readings from the current state and future state of the network and provide an optimal solution for the next course of action.

Eaddie based her methodology on the development of mechanisms that would make a smart decision on what commodity to route across the network that would bring optimal results. A controller would consider taking a commodity and the characteristics of the file size and priority level along with input of the available bandwidth and CPU speed. The controller then calculated the best encryption algorithm for the commodity maximizing the information flow through a network.

There were four steps to the development of the controller, step one involved developing the cryptographic tool that would produce results of how long each commodity would take to be encrypted and decrypted. Step two involved interacting with **Matlab**<sup>®</sup> [9] in the form of using the data from step one and interpolating it. Step three was the development of the controller, the controller would optimize the data from the previous step. The last step, step four was the simulation phase with ns2 [13].

Step one involved creating a user interface to interact and collect the statistical data from GNU Privacy Guard (GnuPG), an open source cryptographic software. GnuPG is a software tool that allows to encrypt and sign your data and communication, features a versatile key management system [7]. GnuPG also has decryption features as well. GnuPG support several encryption and decryption algorithms to secure and unlock your data or communications. Some preliminary definitions regarding encryption terms are as follows:

**Key** A randomly generated set of numbers or characters that is used to encrypt or decrypt information [12]

**Public Key** Also referred to as asymmetric encryption, each party has a private key known only to themselves, and a public key known by anyone, each encryption or decryption process requires at least one public key and one private key. [12]

**Private Key** Also referred to as symmetric encryption, it requires all parties that are communicating to share a common key [12]

**Key Length** Also known as key size and the number of bits, string of bits of the particular parameter

**Block size** The length of the fixed string of bits used in block ciphers

The following is a brief overview on the types of algorithms tested during this step.

**ElGamal** An asymmetric key encryption algorithm for public key algorithm which is based upon the Diffie-Hellman key agreement and uses randomization in the encryption process, further encryption is based upon the intractability of the discrete logarithm problem [11]

**RSA** A public key encryption scheme named after inventors R. Rivest, A. Shamir, and L. Adleman. Widely used to provide both secrecy and digital signatures; encryption is based upon intractability of the integer factorization problem [11]

**AES** Advanced Encryption Standard is a symmetric block cipher that has capability of keys 128, 192, 256 bits to encrypt and block of 128 bits to decrypt. It's based upon the Rijndael algorithm where there is substitution linear transformation [6]

**TwoFish** A symmetric key block cipher with block sizes of 128 bits and key sizes up to 256 bits. Also based upon a Feistel network with a bijective function [16]

**BlowFish** A symmetric block cipher that takes variable length keys from 32-448 bits and based upon a Feistel network iterating a simple encryption function 16 times. [12]

**3DES** Triple-Data Encryption Standard. Uses the DES block cipher encryption that encrypts data three times and uses a different key for at least one of the three passes giving a cumulative key size of 112-168 bits [12]

**CAST5** A symmetric block cipher using a 12 or 16 round Feistel network with 64-bit block size and a key size between 40 to 128 bits. [1]

A gpgTester program was developed that was written in C++ served as a front-end to GNU Privacy Guard. The gpgTester was created to test several cryptographic

algorithms to gather the statistical data on the outcome of using commodities with sizes of one megabyte(MB), sizes in the range of five MB, in increments of five, to 100 MB, and also sizes in the range of 200 MB with increments of 100, to 1000 MB. The `gpgTester` would make system calls to the GNU Privacy Guard to choose from system and public key algorithms, ElGamal, RSA, AES, AES192, AES256, TwoFish, BlowFish, 3DES, and CAST5 for collecting the encryption and decryption time, the resulting files size after encryption, and also the time it took to compression the commodity. The resulting statistical data was then forwarded to step two, for `Matlab`<sup>®</sup> [9] calculations. Eaddie states that with `gpgTester`, the public key algorithm is chosen with another algorithm when a system call to GnuPG occurs. The choices are RSA with RSA, RSA w/Elg-E, DSA with RSA, or DSA w/Elg-E. Regardless of which algorithm is chosen first, it is the second algorithm that is actually used for encryption. The first algorithm is strictly for signing and is not factored into the test results nor used in any timing [21].

During step two, `Matlab`<sup>®</sup> [9] was integrated. Once the `gpgTester` has gathered data from a group of commodities, that data was then read into `Matlab`<sup>®</sup> [9] to interpolate data to produce data points for curve fitting. An encryption algorithm `Matlab`<sup>®</sup> [9] function was created for six encryption algorithms. An interpolation was required to determine the output for any given file size based upon the encryption algorithm, the compression function, the key size, and randomness of the file [21]. Cubic spline interpolation using polynomials because of a practical feature of cubic splines is that they minimized the oscillations in the fit between the interpolating points cubic spline equation. There were two constraints to ensure that data outside the tested data ranges were not included. One restraint was using specific key sizes of 1024 or 128 if compression of the commodity was required, secondly, if the commodity was not generated randomly, then the commodity size limit was 32 MB. `Matlab`<sup>®</sup> [9] files `runCryptGrav` was created to determine the file size from the encryption and the time that it took to encrypt the file, this was to optimize the data. The output from

this function was a determination on what could be sent given the current bandwidth and current CPU speed and requested encryption algorithm.

Further, three **Matlab**<sup>®</sup> [9] functions were created `runCryptGrav`, `runCryptGrav2`, `runCryptGrav3` which performed different interpolation functions based upon certain statistical data collected from the previous steps. I will use the function `runCryptGrav3` in the Decisive Routing and Admission Control Constraints framework, this function is what Eaddie used to run her ns2 [13] simulations, this will be explicitly defined in the methodology section. The function, `runCryptGrav3`, was a combination of the functions `runCryptGrav` and `runCryptGrav2`. It used a cubic spline function to interpolate data points, its input parameters were the data that was collected by the `gpgTester` program, and it also included the “matrix of communication commodities which include the commodity goodness values (priorities) and file sizes.” The results from this function execution is an output matrix in preparation of the input to the next step, the controller.

Once the data came from step two, the **Matlab**<sup>®</sup> [9] Controller read in the data and performs the calculations. Three controllers were created to test different types of data. To set up testing for the first two controllers, data input was randomly generated using a gamma random distribution, see Equation 2.4, to generate random file sizes in the range of one MB to 100MB and the priority levels was then distributed randomly by a density function in the range of one to 100, see Equation 2.5. This generated data would then serve as the input matrix for the controllers. The first controller read in an input file which had the available bandwidth, available CPU, the number of commodities that need to be sent, security level and performance level as listed in Table 2.1 and determined the best encryption algorithm that would give the randomly generated list of commodities that would maximize the available resources. A particular enhancement will be to test whether the available bandwidth can come from the predictions of the Kalman filter and make this a dynamic input.

Table 2.1: Sample Input File for Encryption Controller 1 [21]

Available Bandwidth (MB)	Available CPU (sec)	Number of Commodities	Security Level	Performance Level
120	3000	20	4	3
100	2400	30	5	2
90	2000	40	1	1
200	1900	57	2	1
50	4000	60	3	2
300	6000	70	4	2
70	3400	83	5	3
75	2000	90	3	3
30	1900	100	2	2
20	3000	26	1	2

$$g(t) = \frac{\lambda^\alpha}{\Gamma(\alpha)} t^{\alpha-1} e^{-\lambda t}, t \geq 0$$

where

(2.4)

$$\Gamma(x) = \int_0^\infty u^{x-1} e^{-u} du, x > 0$$

$$f(x) = \frac{1}{n}$$
(2.5)

A derived security function would then take the security level and performance level associated with the commodity and produce the encryption algorithm and the key size to use in sending these commodities. The controller would then take these results and call the runCryptGrav function from step two, which would then act as the decision making tool as to what could be sent.

A conceptual design of the options for the controller is given by Figure 2.9. There is a security setting dial, that relates to the level of security. The level of security would mimic the algorithms the customer would like to use. The performance level relates to customer defined level. These options can be integrated with the above concept of the goal lattice to become more tailored to mission objectives and goals.



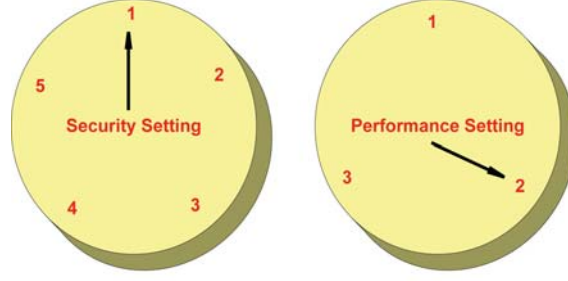


Figure 2.9: Conceptual dial settings for controller one scenario

The numbers on the security dial represent the levels of security a commodity may have. (5) High security, (4) Medium High, (3) Medium, (2) Medium Low, (1) Low. The performance level could be changed to (3) High, (2) Medium, (1) low [21].

The controller is used as the decision making device. It looks at what commodities need to be sent, the security levels, performance levels, and available resources and optimize the results. Optimal decisions are based upon which encryption scheme would be the best to use, looks for the maximum number of commodities to send based upon available bandwidth resources, looks for the lowest amount of time it takes to encrypt the data, and the largest output file size.

Another controller was developed and tested using binary integer programming. The binary choice was zero if the commodity was not sent or a one if the commodity was sent. It did not take in consideration any security or performance levels, it ran the **Matlab**<sup>®</sup> [9] branch-and-bound algorithm for all seven encryption algorithm choices for each commodity to determine the optimal solution. According to Eaddie, the objective was to maximize  $f'x$  such that  $Ax \leq b$ . Further explanation will be included in the methodology section because the controller I'm extending in my research is the combination of the first two controllers using the idea of security and performance levels with binary integer programming.

### III. Decisive Routing and Admission Control Methodology

This chapter is the methodology regarding the Decisive Routing and Admission Control According to Quality of Service Constraints Framework. The framework is basically acting as the glue that is bringing together three distinct frameworks capitalizing on the strength of the combination and cohesion as a whole.

#### 3.1 Overview

Meeting the five baseline system requirements mentioned in the literature review, which were to have:

1. The ability of the system to consider the mission importance of information and make decisive decision accordingly.
2. The ability of the system to be fluid in the tradeoff between making decisive decision based upon network resources and/or mission objectives.
3. The ability of the system to receive all information based upon a tasking order including its parameters/characteristics then determine what is the best solution of what data should be transported from source to destination.
4. The ability of the system to handle and adapt to reoccurring changes in bandwidth and traffic fluctuations.
5. The ability of the system to negotiate the best solution for multiple mission objectives given one distributed System.

Decisive Routing according to Quality of Service Constraints Framework will strive to meet those requirements by combining and extending some of the features and capabilities of the distinct frameworks of Stochastic Estimation and Control of Queues within a Computer Network, Creating an Agent Based Framework to Maximize Information Utility, and Dialable Cryptography for Wireless Networks. The decisive routing framework would be a multilayered approach that lies at the node level in a global network. The smarts of the system will rely on agents that reside at that level as well.

A Network Tasking Order or requirement to send data from a sender to a receiver is what would possibly be used to kick off a chain of events. Knowing what the customer wants to send and when, they can get a sight picture of the network by turning on the Kalman filter prediction module to gather data from the sending or receiving nodes of interest. Further if encryption is involved, by feeding the data parameters and the available encryption algorithms into the encryption module you can get a sight picture of what is the best algorithm to use for optimal end-to-end service. Also at the node of interest the Hybrid Communication Agent [14] will reside. This will give a reactive stance when the network starts to become congested, giving the ability for congestion management and admission control. Due to the congestion of network traffic, the response can trigger the relevant behaviors of the agent to again optimize traffic flow from the nodes within the customer's control. The Kalman filter prediction module can be used in all instances and service, that way a better overall global sight picture can be determined and precise decisions are made in each circumstance.

### ***3.2 Network Prediction Module***

The Network Prediction module will gather pertinent data from sensory type nodes and make futuristic depictions of the network state based upon the data gathered and the use of Extended Kalman filter calculations. Given the ability to know in advance what the state of the network is, some decisions can be made as to what would be the most optimal route for data to travel from the sender to receiver in a manner that is acceptable and optimal.

A Kalman filter is being used to predict past, present, and future states of the network queue. According to [22], the filter is essentially a set of equations that executes a predictor-corrector type estimator that is optimal in that it minimizes the estimated error covariance when some presumed conditions are met. The original design of the stochastic estimation research was focused on estimating the present state of a computer network, given parameters in a controlled system configuration and

used the extended Kalman filter, this linearized about the current mean and covariance. The Decisive Routing According to Quality of Service Constraints extends this framework by modifications of the Kalman filter calculations to predict future states of the queue in a computer network. This extends the functionality by implementing a predictability of the size of the node's queue at some time in the future. The enhancement results in the ability to meet the above stated requirements, to be decisive in decision making of handling information based upon the state of the network.

The code was ported from a Windows based system where the simulation experiments was accomplished in OPNET<sup>®</sup> to a Linux based system where the simulation experiments are performed in Network Simulator Version 2, ns2 [13]. Several class structures were created to build the relationship with the Hybrid Communications Agent based framework. This Network Prediction module, proposed, utilizes the Kalman filter calculations to use the present data to forecast future results, it is written in Matlab<sup>®</sup> [9] and executed through C/C++ Matlab<sup>®</sup> [9] engine as well as extended to Tool Command Language (TCL), so ns2 [13] can run simulations.

The following Network Prediction module class diagram, in Figure 3.1 depicts the relationship with the HANC agent. The diagram is meant to show the interactions and relationship between the classes also the functions that are related. The level started is at a queue level and goes up in a hierarchial manner. Each queue in a computer network has the capability of functioning as a queue of Kalman filter type. The behavior and functionality of the Kalman filter queue takes on the behavior of a queue but with some added characteristics and additional behaviors. The routing scheme chosen for this research was drop tail. When a queue becomes overloaded, it drops the tail end or last data packets to arrive, see Figure 3.2. When a Kalman Filter Drop Tail queue is created, it builds a "has a" relationship with the Kalman Filter class. The added behavior of a Kalman filter Drop Tail queue is to get the location of the current packet in order to determine the location of the queue, additionally an update method that uses the instance of the Kalman Filter class to update the packet data received at the queue. Other behavior and housekeeping methods created were

to query for the unique queue identifier and query to see if the node has a Kalman filter.

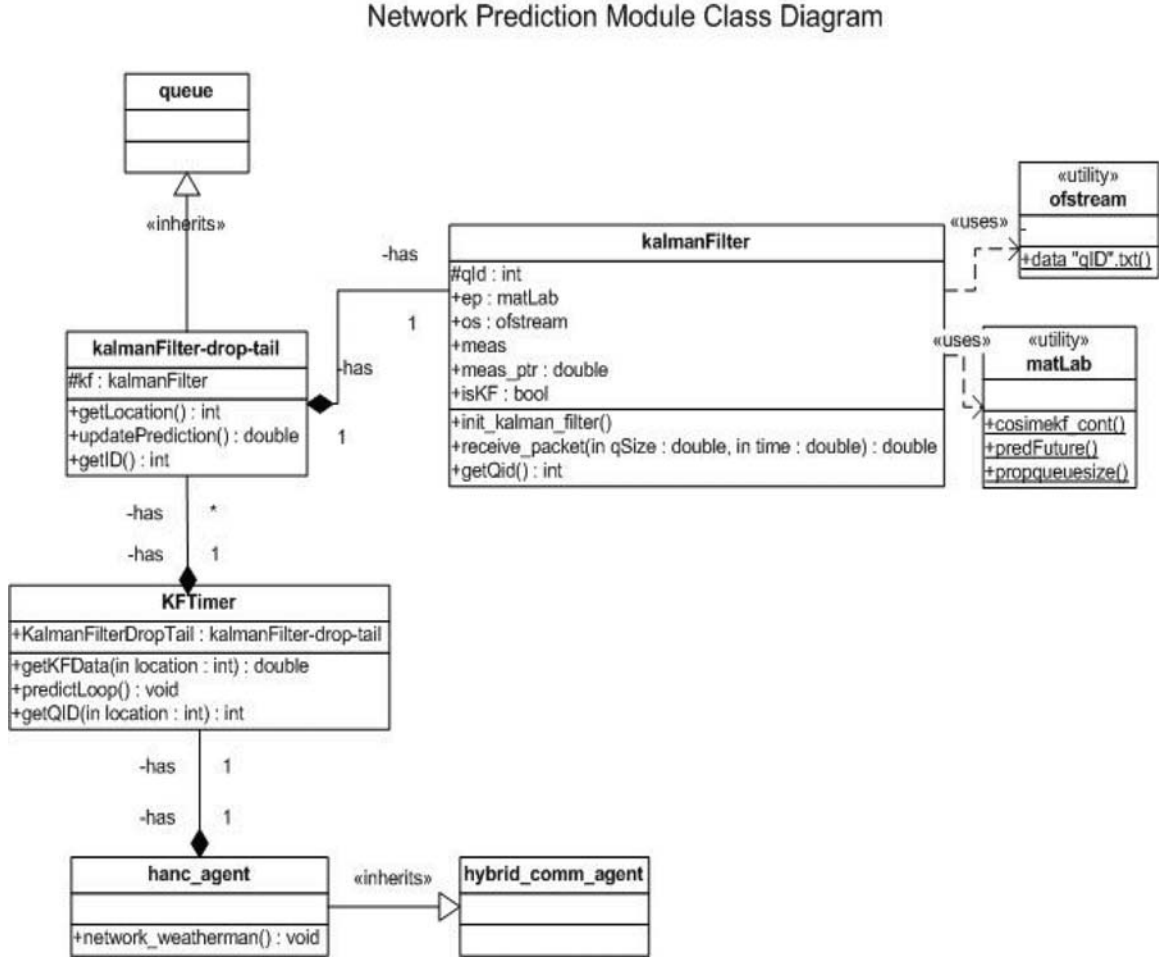


Figure 3.1: Network Prediction Module and Hybrid Agent Network Control Module Unified Modeling Language Class diagram of the relationships between the Network Weatherman structure and the Hybrid Communications Agent structure. As each queue is instantiated to be an Kalman filter Drop Tail queue, it takes on the behaviors of a queue and added behaviors to monitor packets received in order to calculate estimation data.

The original designed controller is not being used in this research, the values are being controlled by the Kalman Filter class file during initialization. The values in the class can be changed to be utilized as the controller for the filter on each node. The Kalman Filter class file is where the integration with Matlab® [9] takes place. This class acts as the interface between the interworking of the queue and Matlab® [9]. When a Kalman filter queue is instantiated it creates a Kalman Filter class where

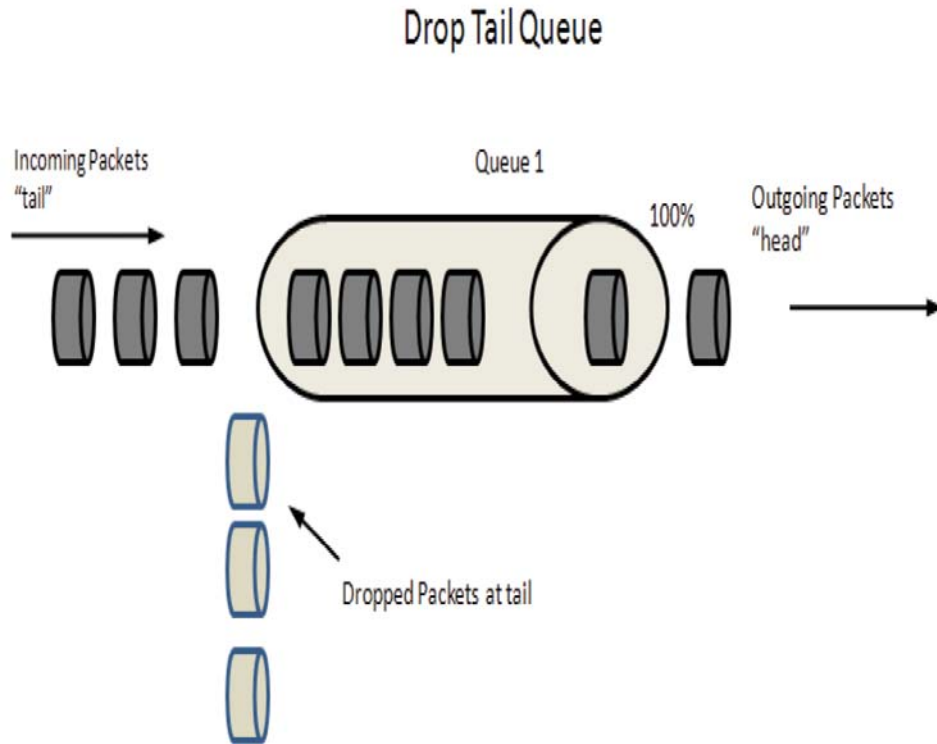


Figure 3.2: Drop Tail Queue Representation

Drop Tail Queue representation, where data packets are arriving at the tail end of Queue 1. Since Queue 1 is at 100% capacity, data packets are being dropped at the tail end.

all variables for the **Matlab**<sup>®</sup> [9] engine are initialized and a unique queue identifier is created. The class receives the call from the Kalman Filter Drop Tail queue every time a packet is received at its queue. Queue size and packet arrival rate are the two states examined by the filter. Given the current queue size and the time as input, the **Matlab**<sup>®</sup> [9] engine runs the extended Kalman filter calculations. When the calculations are finished, the results are dumped into a text file, as well as returned to the calling program. As shown in Figure 3.1 where the Kalman filter class “uses” the **Matlab**<sup>®</sup> [9] engine, where three files are used in this computation. One file performs the queue estimation, one calculate Marcum’s Q-functions, and the other one performs the prediction.

The filter estimates a process by using a form of feedback control, the filter estimates the process state at some time and obtains feedback in the form of noisy

measurements [22], See Figures 2.4 and 2.5 from Section 2.5 as a reference. There are a couple of steps in the estimation and prediction process and the mathematical equations fall into the perspective process groups. One is a time propagation step and the other is the measurement update step in determining the Kalman filter values. Marcum's Q-functions are used as the Kalman filter model basis and occurs in the propagation step. The Q-functions are chosen to simplify the calculations of determining the queue size instead of using the probability density function Equation 3.1 from [20] which is the combination of probability generating functions, partial differential equations and Laplace transforms.

$$\begin{aligned}
p_n(t) = e^{-(\lambda+\mu)t} & \left[ \left( \frac{\lambda}{\mu} \right)^{(n-n_0)/2} I_{n-n_0}(2t\sqrt{\lambda\mu}) \right. \\
& + \left( \frac{\lambda}{\mu} \right)^{(n-n_0-1)/2} I_{n+n_0+1}(2t\sqrt{\lambda\mu}) \\
& \left. + \left( 1 - \frac{\lambda}{\mu} \right) \left( \frac{\lambda}{\mu} \right)^n \sum_{j=n+n_0+2}^{\infty} \left( \frac{\lambda}{\mu} \right)^{-j/2} I_j(2t\sqrt{\lambda\mu}) \right]
\end{aligned} \tag{3.1}$$

For all  $n \geq 0$ , where  $I$  is the infinite series for the modified Bessel function of the first kind given by  $I_n(y) = \sum_{k=0}^{\infty} \frac{(y/2)^{n+2k}}{k!(n+k)!} (n > -1)$

The use of the modified Bessel function of first kind zero ( $0$ ) order and modified Bessel function of first kind of order ( $m-1$ ) are also used in the mathematical processing of Marcum's basic structure of the Q-functions, which are shown below:

$$marcum(a, b)$$

$$Q(a, b) = \int_0^b x \exp\left(-\frac{x^2 + a^2}{2}\right) I_0(ax) dx$$

$$marcum(a, b, m)$$

$$Q(a, b) = \frac{1}{a^{m-1}} \int_0^b x^m \exp\left(-\frac{x^2 + a^2}{2}\right) I_{m-1}(ax) dx$$

Replacing the parameters would simplify into Equations 2.1 taken from Section 2.5. Queue prediction involves taking the present value and looking at some variable time in the future to calculate the results of the queue size. The prediction considers error covariance, and uncertainty with the prediction. Queue prediction for future state was created by modifying queue prediction of present state, using a propagation and an update step.

The Network Weatherman is built upon a discrete-time system model as in Equations 3.2 and 3.3 [20]. If we let  $n$  represent the different states and  $m$  represent the different measurements, it follows that  $x(t_i)$  is illustrating the system state at time  $t_i$  and that  $\phi$  represents the changing from one state to the next from time  $t_{i-1}$  to  $t_i$ . Both of which are  $n$ -dimensional vectors. Lastly,  $w_d$  is unknown dynamic noise of the system, an  $n$ -dimensional function containing discrete-time white Gaussain noise of zero mean and covariance kernel and  $Q_d$  is the  $n$ -by- $n$  matrix expressing the covariance of it.

$$x(t_i) = \phi(t_i, t_{i-1}, x(t_{i-1})) + w_d(t_i) \quad (3.2)$$

$$E\{w_d(t_i)w_d^T(t_j)\} = \begin{cases} Q_d & t_i = t_j \\ 0 & t_i \neq t_j \end{cases} \quad (3.3)$$

Next, the Network Weatherman is built upon a discrete-time measurement model as in Equations 3.4 and 3.5 [20].  $z$  carries the role of a  $m$ -dimensional measurement vector while  $H$  is an  $m$ -by- $n$ -dimensional measurement matrix. Again noise is accounted for in the measurement because of an uncertainty factor and is given by  $v$  an  $m$ -dimensional vector same as  $w_d$  and has a covariance  $R$  an  $m$ -by- $m$  matrix.

$$z(t_i) = Hx(t_i) + v(t_i) \quad (3.4)$$



$$E\{v(t_i)v^T(t_j)\} = \begin{cases} R & t_i = t_j \\ 0 & t_i \neq t_j \end{cases} \quad (3.5)$$

Finally, the Network Prediction module state transition is addressed. The state transitions is contained in a  $n$ -by- $n$  matrix after execution of the partial derivative Equation 3.6 [20]. The estimated value of the state is given by the Kalman filter at time  $t_{i-1}$  after the measurement update is  $\hat{x}(t_{i-1}^+)$ .

$$\Phi_{ij}(t_i, t_{i-1}) = \frac{\partial \phi_i}{\partial x_j} \Big|_{x=\hat{x}(t_{i-1}^+)} \quad (3.6)$$

However, a two-sided difference equation is used because  $\phi$  is not continuous. Therefore during the time and queue propagation step the following equations occurs via the **Matlab**<sup>®</sup> [9] engine.

$$\Phi_{11} = \frac{\phi_1 \left[ x + \begin{bmatrix} 1 \\ 0 \end{bmatrix} \right] - \phi_1 \left[ x - \begin{bmatrix} 1 \\ 0 \end{bmatrix} \right]}{2} \Big|_{x=\hat{x}(t_{i-1}^+)} \quad (3.7)$$

where  $\Delta x_1$  is chosen to be 1 due to the smallest perturbation allowed because the queue size must be an integer.

$$\Phi_{12} = \frac{\phi_1 \left[ x + \begin{bmatrix} 0 \\ \frac{x_2}{100} \end{bmatrix} \right] - \phi_1 \left[ x - \begin{bmatrix} 0 \\ \frac{x_2}{100} \end{bmatrix} \right]}{2 \frac{x_2}{100}} \Big|_{x=\hat{x}(t_{i-1}^+)} \quad (3.8)$$

$$\Phi_{21} = 0, \Phi_{22} = 1 \quad (3.9)$$

where  $\Delta x_2$  is chosen to be  $\frac{x_2}{100}$  to provide a scaled perturbation that was two orders of magnitude smaller than the state, this value does not have the requirement to be an integer.

Once the transition states are approximated, the following equations [20] are being called, where superscript “-” is before the measurement update and superscript “+” is after the measurement update. The Q-functions are calculated, the expected value of the number of packets in the queue as a function of time is calculated from Equations 2.2 and 2.3 from Section 2.5 and incorporated into the additional equations.

The estimate  $\hat{x}$  and covariance  $\mathbf{P}$  are propagated from  $t_{i-1}$  to  $t_i$  by

$$\hat{x}(t_i^-) = \phi(t_i, t_{i-1}, \hat{x}(t_{i-1}^+))$$

$$\mathbf{P}(t_i^-) = \Phi(t_i, t_{i-1})\mathbf{P}(t_{i-1}^+)\Phi^T(t_i, t_{i-1}) + Q_d$$

Once the propagation step is complete, the measurements gets updated. One of the added benefits of using Kalman filter processing during the update step is the consideration of the typically noisy nature of measurements. During this step, noise is determined by using Cholesky Factorization [20] and a normally distributed random number. The Cholesky factorization parameter is received from the Kalman filter class and can be changed or tweaked as desired.

In the update step, the measurements are then incorporated through the following equations [20], where  $K$  is the Kalman filter gain presenting an  $n$ -by- $m$  matrix.

$$\mathbf{K}(t_i) = \mathbf{P}(t_i^-)\mathbf{H}^T[\mathbf{H}\mathbf{P}(t_i^-)\mathbf{H}^T + \mathbf{R}]^{-1}$$

$$\hat{x}(t_i^+) = \hat{x}(t_i^-) + \mathbf{K}(t_i)[\mathbf{z}(t_i) - \mathbf{H}\hat{x}(t_i^-)]$$

$$\mathbf{P}(t_i^+) = \mathbf{P}(t_i^-) - \mathbf{K}(t_i)\mathbf{H}\mathbf{P}(t_i^-)$$

A summary of the operation is shown in Figure 3.3, modified from [22]. A brief description of the variables are  $\mathbf{Q}_d$  is an  $n$ -by- $n$  matrix representing the covariance of  $\mathbf{w}_d$ , which symbolize system dynamic noise.  $\mathbf{z}$  is the  $m$ -dimensional measurement vector and  $\mathbf{H}$  is the  $m$ -by- $n$  measurement matrix for the states.  $\mathbf{K}$  is Kalman filter gain  $n$ -by- $m$  matrix.  $\mathbf{P}$  is an  $n$ -by- $n$  matrix for covariance with  $\hat{x}$ , and  $\mathbf{R}$  is an  $m$ -by- $m$  matrix, covariance of  $v$ , the measurement noise.  $\phi$  and  $\Phi$  refers to the transition of the state of the system.

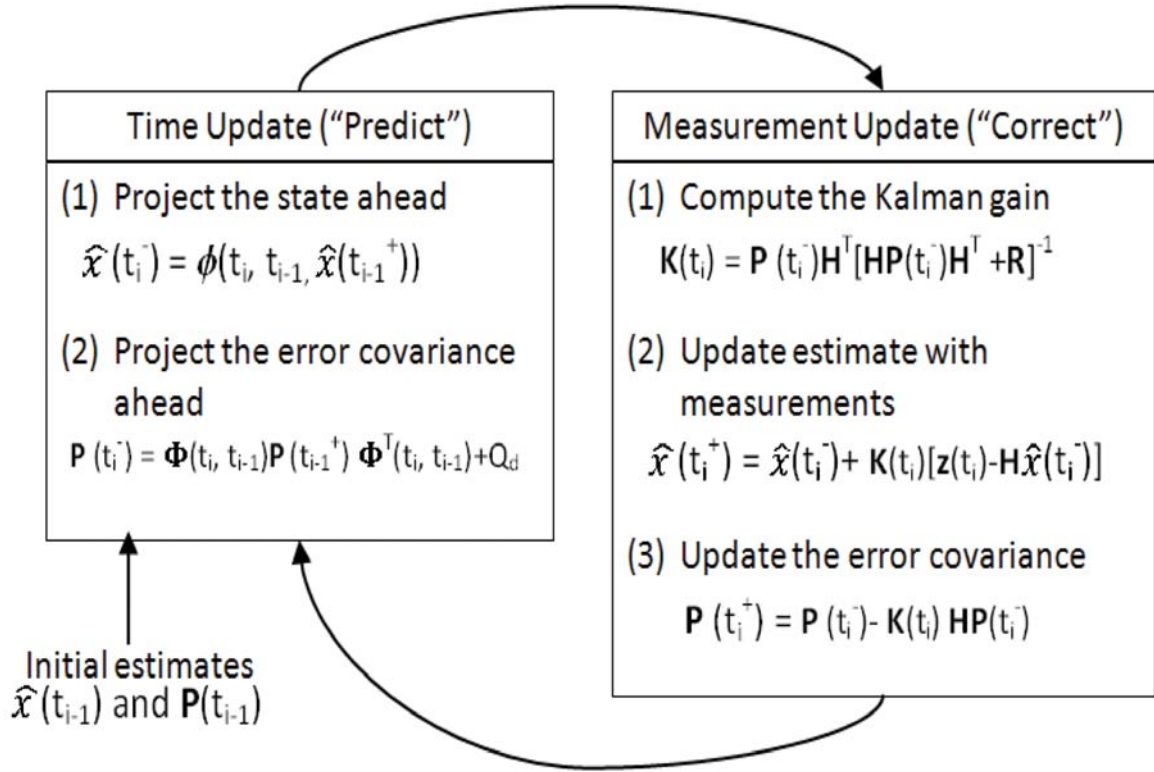


Figure 3.3: Network Prediction Module and Hybrid Agent Network Control Module

A brief description of the variables are  $\mathbf{Q}_d$  is an  $n$  by  $n$  matrix representing the covariance of  $\mathbf{w}_d$ , which symbolize system dynamic noise.  $\mathbf{z}$  is the  $m$ -dimensional measurement vector and  $\mathbf{H}$  is the  $m$  by  $n$  measurement matrix for the states.  $\mathbf{K}$  is Kalman filter gain  $n$  by  $m$  matrix.  $\mathbf{P}$  is an  $n$  by  $n$  matrix for covariance with  $\hat{x}$ , and  $\mathbf{R}$  is an  $m$  by  $m$  matrix, covariance of  $v$ , the measurement noise.  $\phi$  and  $\Phi$  refers to the transition of the state of the system.

The predicted data from the run of the Kalman filter class will then be used to calculate a delta in capturing the average size of the queue. Comparing the previous delta with a current delta will give a trend of the behavior of the queue and deterministic decisions can be made based upon that trend. If the queue is seemingly

increasing in size, then the HANC Agent can be told to send messages to its neighbors to increase the traffic flow because it can handle more traffic. If the queue is seemingly decreasing in size, then HANC agent can be told to send messages to decrease the traffic flow because it cannot handle high message traffic.

The KFTimer class builds the bridge from the Kalman filter drop tail queue to the HANC Agent. The method takes sensory readings at set times during the simulation and run the prediction loop code. The timing can be modified to take reading according to the customer specifications. Once the sensory data has been collected, a global sight picture can be realized on what the state of the queue will be in the future.

### ***3.3 Encryption Optimization Module***

This module is based upon the Dialable Cryptography for Wireless Networks controller three 2.7. It is a combination of controller one that integrates security levels and performance levels in the optimization decisions as well as binary integer programming from controller two. The class diagram in Figure 3.4 indicates the relationship with HANC and the Encryption module. The diagram shows that encryption requests are called from the HANC agent, and that request then calls the proper `Matlab`<sup>®</sup> [9] function to interpret results from the given parameters. A detailed explanation of the parameters and matrices are included because of the integration of the three framework modules. The current module design is that input is taken from a function that will take sample hard coded data and then put that data in a correct format and matrix for `Matlab`<sup>®</sup> [9] to use and interpolate the results. This sample data would include the available bandwidth, the available CPU speed, a commodity number, the file size of the commodity, the priority level, the security level, and the performance level. A sample of what file will be created and tested is in Table 3.1. An extension for this module is to make the inputs more dynamic rather than the results of hard coded data. Another test to be conducted is the available bandwidth parameter. A

Table 3.1: Sample Input File for Encryption Optimization Module [21]

Available Bandwidth (MB)	Available CPU (seconds)	Placeholder		
100	2400	0	0	0
Commodity	File Size (MB)	Priority Level	Security Level	Performance Level
1	10	60	3	1
2	15	28	2	2
3	30	90	3	3
4	5	30	2	1
5	50	15	4	2
6	60	75	1	1
7	3	90	5	3
8	22	49	2	2
9	20	74	2	3
10	44	38	5	2

simulation will be set up to receive results from the network predictor and perform a encryption optimization scheme based upon received results.

Therefore, based upon this sample input, the security level and performance level was matched up with a selection of appropriate encryption algorithms according to Table 3.2. Each level tested three encryption algorithms at that level. Security level's ranged from one to five where five was the highest. Performance level ranged from one to three where three was the highest.

Then the resulting data is fed as input into the function to gain statistical data of encryption file size and encryption time. Once that is complete **Matlab**<sup>®</sup> [9] is then called and given these inputs in a matrix form to perform binary integer programming. **Matlab**<sup>®</sup> [9] along with GNU Linear Programming Kit (glpk) is used to solve the optimization of which commodity to send with which encryption algorithm. The parameter inputs are calculated to optimally solve  $Ax \leq b$ . Each commodity has an option to be sent or not sent thereby creating the binary of 1 or 0.

$$x_j = \begin{cases} 1 \\ 0 \end{cases}$$

## Encryption Optimization Module Class Diagram

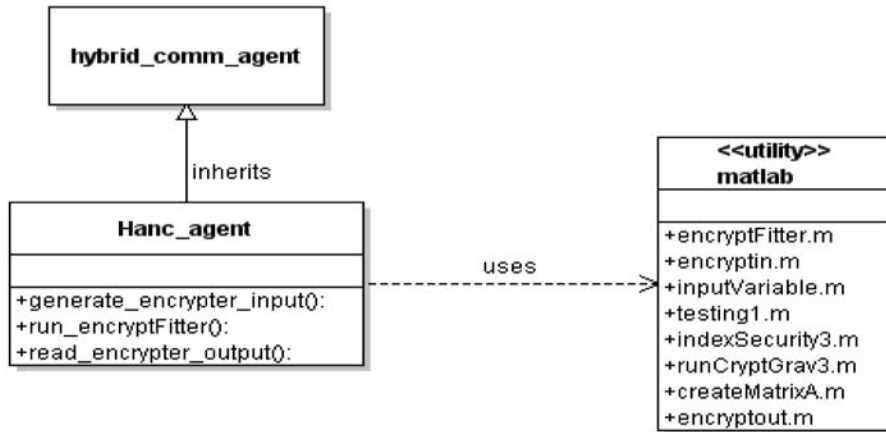


Figure 3.4: Encryption Optimization Module and Hybrid Agent Network Control Module Unified Modeling Language Class diagram of the relationship between the Encryption Optimization Module and the Hybrid Communications Agent structure. The parameters as in Table 3.1 are given and the module interpolates a resulting optimal list of commodities to send and at what encryption algorithm.

Table 3.2: Security and Performance Level Algorithms for Encryption Optimization Module [21]

Security Level	Performance Level	Encryption Scheme &(Key Size)
1	1	RSA (1280); RSA (1536); Elg-E (1280)
1	2	Elg-E (768); Elg-E(1024); RSA (1024)
1	3	3DES, BlowFish, CAST5
2	1	Elg-E(1536); Elg-E(1792); RSA (2304)
2	2	RSA(1792); RSA(2048); Elg-E(2048)
2	3	3DES, BlowFish, CAST5
3	1	Elg-E(2816); Elg-E(2072); RSA (3072)
3	2	RSA(2816); RSA(2560); Elg-E(2560)
3	3	AES; TwoFish, CAST5
4	1	Elg-E(3840); Elg-E(3584); RSA(3840)
4	2	RSA(3328); RSA(3584); Elg-E(3328)
4	3	AES(192); TwoFish; CAST5
5	1	Elg-E(3840); Elg-E(4096); RSA(4096)
5	2	Elg-E(3584); RSA(3840); RSA(3584)
5	3	AES(256); AES(192); TwoFish

Security Level 1 being the lowest, 5 being the highest.

Performance Level 1 being lowest and 3 being the highest.

One parameter is a commodity matrix derived from [21], where each commodity original size, priority, encryption file size, encryption time is enumerated for each encryption algorithm available see below for more details.

A commodity matrix  $c$  for the  $n$  commodities and  $y$  encryption schemes will produce a matrix in the the form of

$$c = \begin{pmatrix} i_1 & p_1 & s_{11} & t_{11} & s_{12} & t_{12} & s_{1y} & t_{1y} \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ i_n & p_n & s_{n1} & t_{n1} & s_{n2} & t_{n2} & s_{ny} & t_{ny} \end{pmatrix}$$

Where the each row in the matrix corresponds to the following

$i_{1...n}$  = input file size of commodity (1... $n$ )

$p_{1...n}$  = priority level of commodity (1... $n$ )

$s_{11...ny}$  = encrypted file size of commodity (1... $n$ ) using encryption scheme (1... $y$ )

$t_{11...ny}$  = encryption time of commodity (1... $n$ ) using encryption scheme (1... $y$ )

A second parameter is a vector of priorities of the enumerated commodities from the  $c$ , where  $n$  is the number of commodities and  $y$  is the number of encryption algorithms. This general coefficient matrix,  $f$ , derived from [21] take on the following form

$$f = \begin{pmatrix} p_1 \\ \cdot \\ \cdot \\ \cdot \\ p_{n*y} \end{pmatrix}$$

The vector of priorities is used as a coefficient matrix to solve  $Z = -f_1x_1 - f_2x_2 - \dots - f_{n*3}x_{n*3}$  equation to maximize the  $x'$ s with the constraint of  $Ax \leq b$  [21].

The constraints for the optimization problem comes from the  $A$  matrix where it defines that at most one encryption algorithm scheme is chosen for the given commodity from the commodity matrix  $c$

$$A = \begin{pmatrix} 1_{11} & 1_{1..y} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1_{21} & 1_{2..y} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1_{31} & 1_{3..y} & 0 & 0 \\ . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & 1_{n1} & 1_{ny} \\ size_{11} & size_{1..y} & size_{21} & size_{2..y} & size_{31} & size_{3..y} & size_{n1} & size_{n..y} \\ time_{11} & time_{1..y} & time_{21} & time_{2..y} & time_{31} & time_{3..y} & time_{n1} & time_{n..y} \end{pmatrix}$$

Where each row represents the commodity and the ability to be encrypted by a scheme indicated by a 1 in each column. The corresponding encrypted file size and encryption time for that particular encryption scheme and commodity is indicated in the last two rows of the matrix.

Another constraint is from the  $b$  matrix where it constrains the at-most value for each commodity meeting the equation  $Ax \leq b$ . Only one encryption scheme can be chosen for a given commodity out of the number of encryption schemes  $y$  given bandwidth (BW) and central processing unit (CPU) values. This vector is derived from [21].

$$b = \begin{pmatrix} 1 \\ 1 \\ . \\ . \\ 1_{n*y} \\ BW \\ CPU \end{pmatrix}$$



Matlab<sup>®</sup> [9] and glpk then populates the matrices to solve the optimal values of  $x$  such that the constraints of allowing only one encryption algorithm to be used, all the encryption times have to be less than the CPU speed, and the sum of all the encrypted file sizes have to be less than the given BW.

### ***3.4 Hybrid Agent Network Control Module***

The overall Hybrid Agent Network Control (HANC) module is reactive in nature. HANC relies on a set of rules in which to apply an appropriate behavior. It becomes reactive based upon information classification, information scheduling, resource allocation, and traffic routing schemes. The results from the Network Prediction module and Encryption Optimization module can be a way to set up rules and behaviors for this module to respond from. A Network Tasking Order or network plan can be used for normal operations. Given network mission objectives and goals, HANC can be programmed to react in a custom-made manner. The type of behaviors, as discussed in section 2.6, are to decisively handle delineated traffic based upon characteristics such as priority, encryption, or status traffic.

This module consists of a sequencer and controller layer. It also has some housekeeping components called coordinator and state. The original design of the sequencer layer was to act as an interface between the coordinator component and the mission objectives and goals for the system. Also the task of the sequencer is to develop a library of behavior sets based upon the type of traffic that is being transported. The idea is to have the coordinator take in the mission objectives and goals and then pass this data over to the sequencer layer where the mission objectives and goals are given a priority or a weight. This priority or weight would then further be used in the delineation of traffic type. An example of a mission objective would be to maximize utilization across the network, therefore, as data or information is being scheduled to be placed on the network, data meeting the goal of maximizing the network can be weighted more with a higher priority or classification so when a decision has to be made, at the router or relay level, whether to drop or keep a piece

of traffic moving, that weight or priority will be a determining factor. This layer from the [14] research did not have much functionality, however, in this research it will be extended to receive inputs from the Network Prediction and Encryption Optimization module to act as mission objectives or goals.

The original design of the controller layer was to react to the environment based upon the information gathered from the sensors and stored in the state component. The controller would control the actions taken in the environment based upon that information. The controller uses a Unified Behavior Framework (UBF). This UBF is adapted from a robotic Three Layer Architecture discussed in [14], it uses voting schemes based upon behaviors and arbitration as to which behavior is chosen. The sequencer layer selects the behavior set library as well as the arbitration behavior sets for the UBF to utilize. Figure 3.5 depicts the relationship and events between the sequencer layer and controller layer. Sensory data is being collected from the network from resources that have HANC, based upon the data collected it is stored in a state component. The sequencer collects the state of the network and matches up behaviors and candidate actions for the library it has built. The composite behaviors are then forward to the arbitration area where the UBF makes a selection from the available choices that would produce the optimal results. The controller then executes the action on the network environment.

The type of behavior or actions the controller generated from the basic design was to meet the mission objective of maximizing utilization. With this mission objective the behaviors or actions that could be taken are as follows:

**Raise Threshold** When the resource is becoming overwhelmed and congested, one action that the resource could take is to tell its neighbors to throttle back on the data that it is sending. The message would cause the neighbors to make a decision as to the traffic it is sending. The differentiation between priority, utility, or value of the data comes in to play here, when higher valued items need to pass, lower value items are dropped.

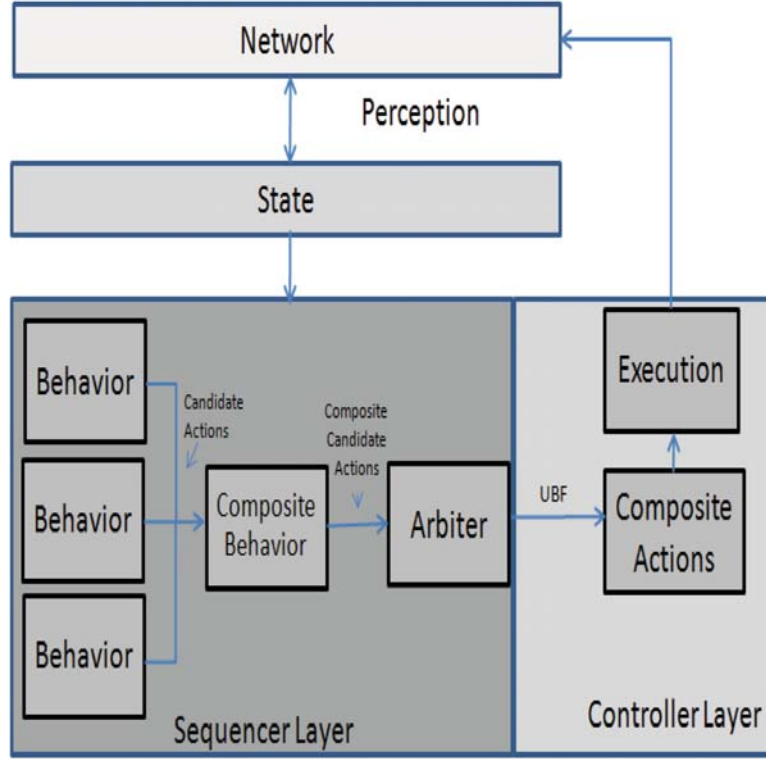


Figure 3.5: Sequencer Layer and Controller Layer

Sensory data is being collected from the network from resources that have HANC, based upon the data collected it is stored in a state component. The sequencer collects the state of the network and matches up behaviors and candidate actions for the library it has built. The composite behaviors are then forward to the arbitration area where the UBF makes a selection from the available choices that would produce the optimal results. The controller than executes the action on the network environment. Figure modified from [14].

**Lower Threshold** When the resource is being underutilized, the resource can send messages to its neighbors requesting more work by allowing lesser valued items to pass. Again the decision making as to admission control at the resource.

**Solve Routing Topology** If a resource have not received any keep alive messages from its neighbor indicating that the neighbor is no longer in service, it sends out a message to update the routes that it can use to keep data moving.

The state component is memory used to hold data gathered or perceived from the network. The original design was for the state component to use preceptors located in the network environment that would collect and capture data. The author states that some sources of information can be gathered from inbound data packets [14].

Inbound data packets can provide information that is maintained in the header such as the identifies, utilities, and size of the data it the arrival rate can be measured as well as an indicator to the state of network. Another source of perception data is the system messages that are generated based upon the behaviors or actions performed. As Pecarina mentions, the act of action messages to raise or to lower a nodes' threshold can be some indicators of environment state. Lastly, knowledge can be collected from the node or resource itself. The node knows its queue size, outbound data rates, and its routing table, these items can be used too. Therefore, these collectable items can be saved in some fashion for use in determining the next or best course of action. As the perception elements change, the state component is updated. As an extension to this component, as the Network Prediction module executes, the collected results can be stored in the state component and used in creating composite behaviors. Another extension is the Encryption Optimization module and use the data stored or update the state based upon its execution. This will strengthen the ability to be decisive in making decisions dynamically thereby meeting the baseline system requirements of being thoughtful, adaptability, and balancing the resources for optimal results.

The last element of the original design for HANC is the coordinator element. The coordinator is the liaison between all resources that house HANC agents. The coordinator ensures system messages are passed to all responsible agents. The coordination of the NTO, mission tasks, mission objectives, as well as the action messages. The author used the coordinator to also send keep alive messages to its neighbors, indicating that the neighbor was still available.

HANC design was enhanced with incorporation of the Network Prediction module for this research. When the state of the system changes based upon the data received from the Network Prediction module, it responds with the corresponding action to either send "raise the threshold" or "lower the threshold" messages to its neighbors. The design has also been enhanced with the functionality of being context aware of the priority of a commodity. When the decision, at the node, needs to be

made regarding whether or not to send a message, the message is only sent if it meets the given criteria.

The incorporation of the Network Prediction module and HANC provides a more proactive stance in making decision given a forecasted state of the network. A delta parameter is being calculated by taking the forecasted queue size readings from the Kalman filter program and storing the values in a circular queue data structure. The queue is currently set to hold five values, as data comes in the next value will be overwritten in a circular fashion. An average is taken of the readings in the queue and stored in variable current delta, it will take up to five readings in order for this value be of good use. As new values as coming in, the last current delta is stored in a variable previous delta and the current delta is the replaced with the updated value.

The difference between the current delta and previous delta are compared, if the current delta is larger, this indicates that the queue is filling up, storage capacity is decreasing. If the current delta is smaller than the previous, the queue storage capacity is becoming larger.

A decision to notify a node's neighbors is based upon the queue storage capacity decreasing to 45% total capacity. If the level is reached, the preemptive throttle back messages goes out to the neighbors time to respond with either stopping traffic, or sending only the most important of its traffic.

## IV. Decisive Routing and Admission Control Results

This chapter describes the results from various simulations and experimentations conducted to test the consolidated Decisive Routing and Admission Control According to Quality of Service Constraints Framework.

### 4.1 *Network Prediction Module*

The Network Prediction Module was tested separately in order to gain the knowledge of the capabilities and the range of parameters to use in order to get the best advantages of combining this module with HANC. Examinations included simulation of a noisy network and how the module would fair against this type of environment. Another examination was determining what range of forecasting could be used with the most accurate results.

*4.1.1 Can Network Prediction Handle Noise?* The first set of experiments involved the Network Prediction Module. The original network prediction was to calculate the present value of the network queue. This code was ported from Microsoft Windows based OPNET<sup>®</sup> to Linux based ns2 [13], a discrete event simulator, and modifications made to calculate the future value of the network queue. A Matlab<sup>®</sup> [9] file was created to interact with the original design to calculate the predicted values. This file is called after the propagation stage and update stage. Within the file, there is a method to propagate based upon a given prediction parameter of how far to look into the future. Furthermore, the prediction variable parameters are the current queue size and arrival rate in which it uses in its calculations. Lastly, this file then performs a error covariance measurement and the uncertainty of the prediction is considered.

Noise in the measurement is calculated by Cholesky Factorization multiplied by a random number generated via Matlab<sup>®</sup> [9]. The actual traffic measurements are noisy because the data rate must be deduced from sample data measurements and that even when the router can be accessed to provide detailed information, noise is

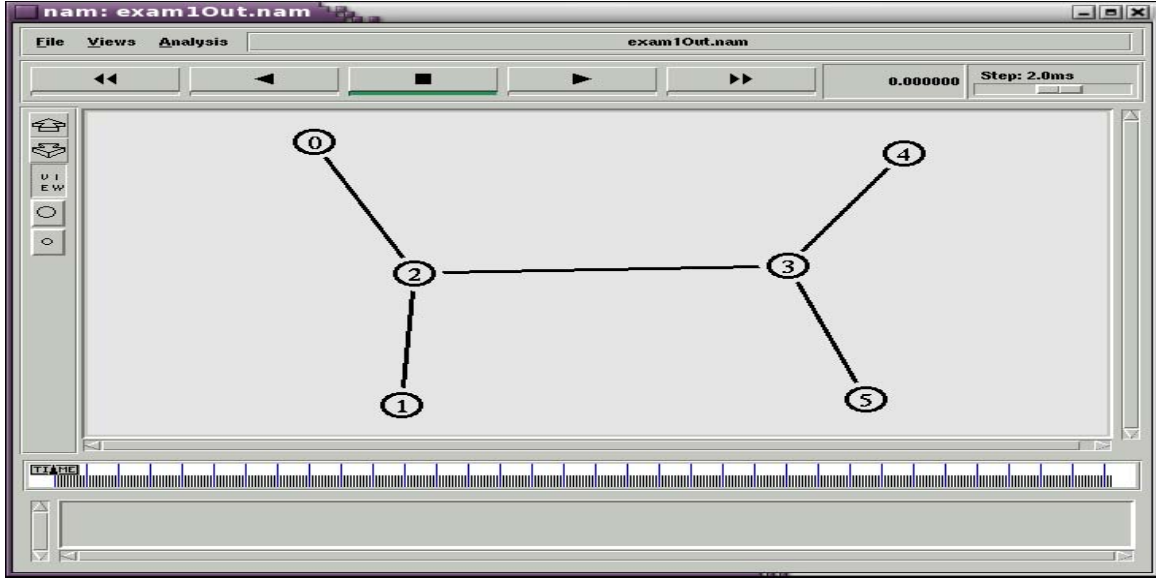


Figure 4.1: Network Prediction Network Layout

Network Prediction Module Network Layout, The Kalman Filter resides at Node 3 and act as a router for three incoming source Nodes 0, 2, and 5. Both Node 3 and Node 4 are the destination nodes.

still present due to delays and errors in the transfer of information [20]. The first set of simulations considers a variation in the amount of noise to the accuracy of prediction results. To set up the simulation, a network was laid out as in Figure 4.1. The figure depicts six Network Nodes. Duplex links of 1 Mega bits per second (Mbs) with 200 millisecond (ms) delay, are setup between Nodes 0 and 2, Nodes 1 and 2, Nodes 3 and 5, and Nodes 2 and 3. The link between Nodes 3 and 4 and 4 and 3 are simplex links with 1 Mbs with 200 ms delay. In order to capture the data of one Kalman filter, simplex links are used. Node 3 is where the Kalman filter resides and acts as a router, passing the data from the sending sources to the destination, Node 4. The other queueing scheme at the other nodes is the drop tail. The queue at Node 3 has a defined limit of 2000, this insures a trend is captured and packet loss is minimized. The sending sources in the network come from Nodes 0, 2, and 5. Nodes 3 and 4 act as the destination or sinks. A variety of traffic is generated via ns2 [13] and simulated to understand the capability of the Kalman filter. Exponential traffic from source 0 is generating from Node 0 to Node 3, with 2200 byte packet size, 5 second burst

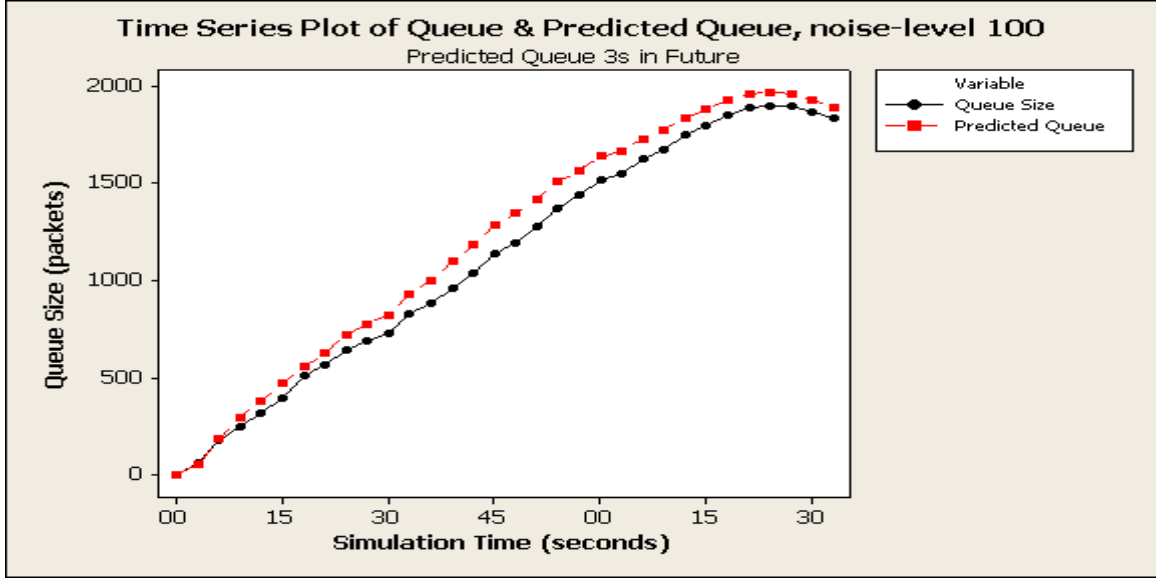


Figure 4.2: Compared Queue Size with Cholesky Factorization 100

Given an input value of 100 to the Cholesky Factorization consideration for noise, the result of the predicted queue size versus the actual queue size vary widely.

time, 3 second idle time, every 600k byte per second. Source 1 is also exponential traffic from Node 2 to Node 4, 3500 byte packet size, 2 second burst time, .5 second idle, and a rate of 450k bytes per second. Source 2, exponential traffic from Node 5 to Node 4, 1700 byte packet size, 3 second burst time, 2.2 second idle, and a rate of 1350k byte per second, There is a TCP connection between Node 1 and Node 3 with FTP also UDP connection between Node 5 and 4 with constant bit rate traffic of 600 byte packet size, and sending rate of .5m bits. The simulation time is set to run for 93 seconds and prediction is timed to take place at every three in the future.

*4.1.1.1 Experiment 1: Noise level 100.* Experiment one trials were conducted to determine the response of the filter with a system of simulated high level noise. Ten runs of the experiment was conducted with Cholesky Factorization value that is passed is 100. The seed for each run was taken from a range of arbitrary values of 129449 through 530390. Figure 4.2 is the result of running the simulation. It depicts a time series plot of actual queue size represented in black and the predicted queue size, represented in red, at a specific time slice of the simulation. The prediction



is being forecasted three seconds into the future, it is calculating what the actual queue size will be based upon the past and current condition of the network, also considering the noise-level. This figure also shows that the predicted values follows the same trend as the actual queue, as the actual queue is increasing or decreasing in packets, so is the foretold value. There are however, variations in the values that are predicted. The minimum percent difference between the actual queue size and the predicted queue size is -14.35% meaning that the forecasted value was below the actual value by as much as 14.35%. The maximum percent difference is 19.63%. The overall mean of the difference between the what is actual and what is calculated is 8.48% according to a 95% confidence interval. Figure 4.3 gives a pictorial view of what the percent difference between the actual queue size and the calculated queue size is at a given time slice of the simulation. Starting around time 72 seconds, the the differences are starting to go below 5%, this is on the account of queue becoming saturated, the queue is at 90% capacity by this time. Based upon a 95% confidence interval, the range of the interval is between 6.15% to 10.84%.

Figure 4.4 indicates with 95% confidence that the mean difference in actual packets versus the predicted queue packets at a given time during the experiment. The overall range of the mean is within the range of -11 to 95. The negative indicates the prediction was underestimated and the positive meaning the prediction overshot the actual value.

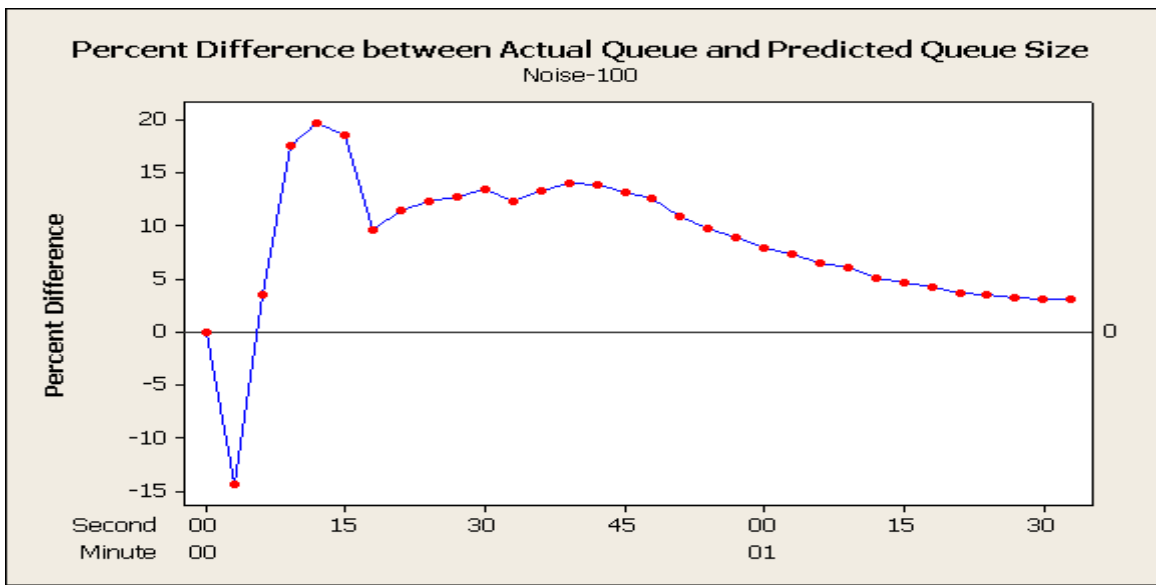


Figure 4.3: Percent Difference between Actual Queue Size and Predicted Queue Size  
During simulation the difference between the two values are calculated. The highest differing value is by 19.63% The values start to normalize when the queue reaches its capacity around 72 seconds.

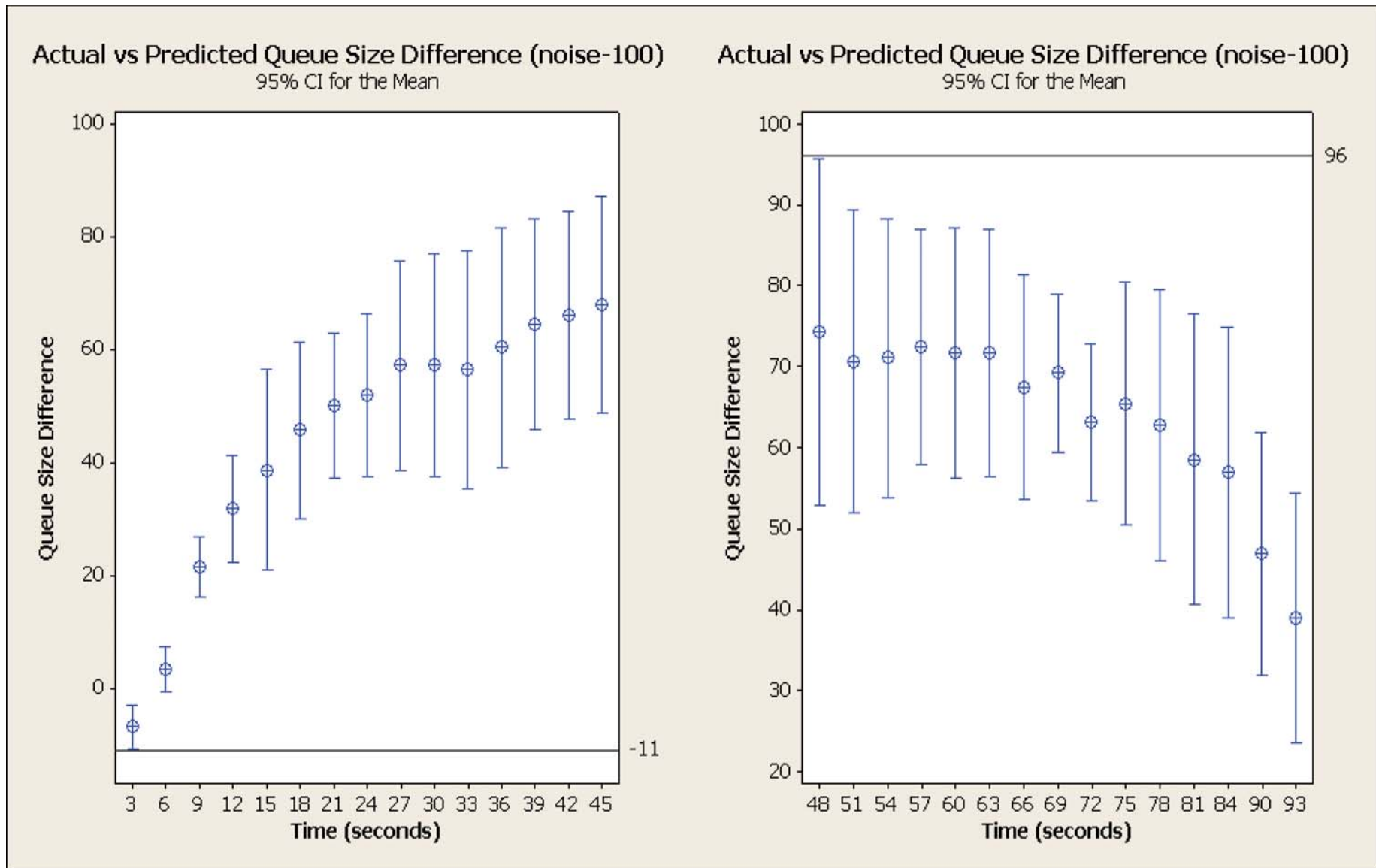


Figure 4.4: 95% Confidence Interval Mean Difference in Queue Size

95% Confidence Interval for the mean value of the difference between the actual queue size and predicted queue size. The noise parameter is set to 100 simulating a very noisy network.

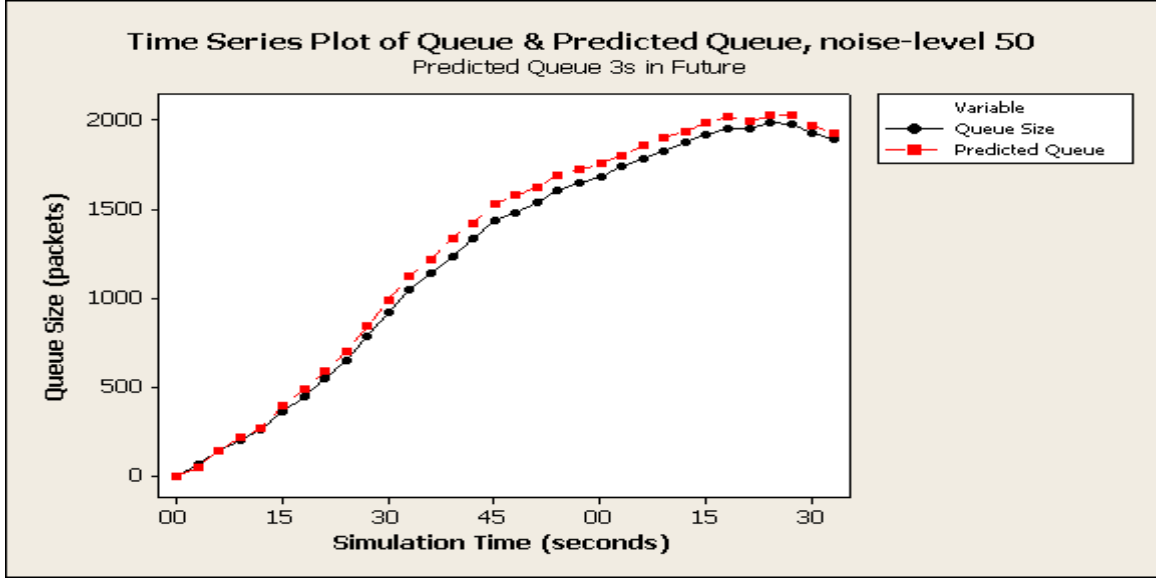


Figure 4.5: Compared Queue Size with Cholesky Factorization 50  
Given an input value of 50 to the Cholesky Factorization consideration for noise, the result of the predicted queue size versus the actual queue size vary somewhat.

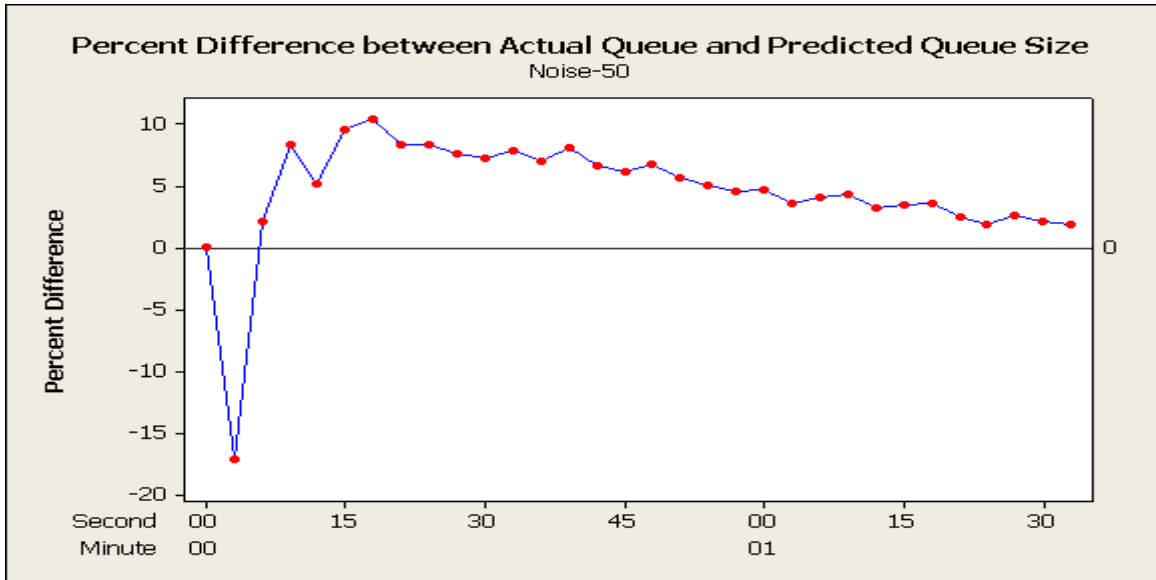


Figure 4.6: Percent Difference between Actual Queue Size and Predicted Queue Size  
During simulation the difference between the two values are calculated. The highest differing value is by 10.44% The values start to normalize when the queue reaches its capacity around 51 seconds.

*4.1.1.2 Experiment 2: Noise level 50.* Experiment two was conducted to determine the response of the Kalman filter with a simulated moderate level of noise. The same scenario is being used as in Level one, but with the Cholesky Fac-

torization value of 50. Ten trials were also run but with a seed value in the range of 1510610 to 9732120. Figure 4.5 is showing the outcome of the simulation. The figure is a time series plot of the actual queue size and predicted queue size, with simulated noise level at 50. The prediction still follows the trend of the actual queue, increasing and decreasing as the actual queue does. The space between the actual value and the forecasted value is closer during this simulation. Using a 95% confidence interval, the mean of the difference between the two plotted lines is 4.53%, with a minimum being -17.16% and maximum difference value of 10.44%. I would consider the -17.16% as an outlier because of the initiation of traffic flow and the random generation of seed for the simulation. The range of the interval is between 2.825% to 6.234%. The difference values also begins to normalize below 5% when the queue is at 75% capacity during this run. Figure 4.6 is capturing the percent difference between the actual queue and predicted queue at specific time slices during the simulation. The overall data values are lower than the first simulation tested, thereby giving an indication of lower noise indicates closer predicted values to the actual values.

Figure 4.7 depicts the experiment with Cholesky Factorization value of 50, and the average mean of the difference between the actual queue size and the forecasted queue at a certain time during the experiment. The confidence interval used was 95%. Therefore with 95% confidence the mean difference of the queue packet size, during the complete experiment, is in the lower range of -20 to a higher range of 117 packets. The negative number indicates the prediction was below the actual queue size and the positive number is the overage of the prediction.

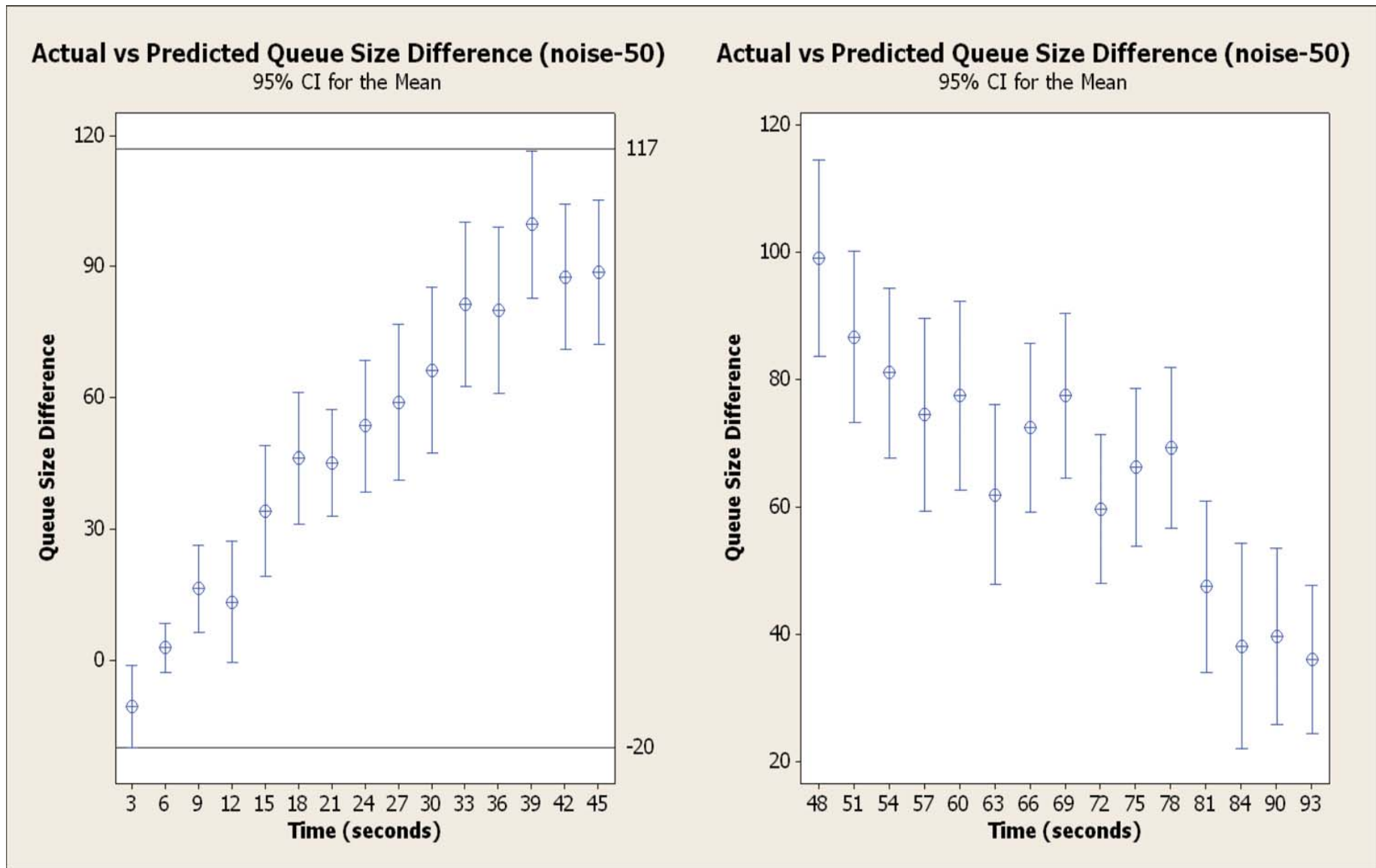


Figure 4.7: 95% Confidence Interval Mean Difference in Queue Size

95% Confidence Interval for the mean value of the difference between the actual queue size and predicted queue size. The noise parameter is set to 50 simulating a mid-range noisy network.

4.1.1.3 *Experiment 3: Noise level 1.* Lastly, the next Cholesky Factorization value used is one. This experiment would represent a network with some noise but normal to low level. Ten trials were conducted with a seed value in the range of 676484 to 859472. Figure 4.8 depicts that the predicted value of the queue size versus the actual value do not vary by much. The predicted value follows the trend and movement of the actual queue plot. The mean value of the difference between the prediction and the actual is only 4.72%. A confidence interval of 95% is in the range of 3.434% to 6.004%. Figure 4.9 shows the minimum difference being -9.62% at the start of the simulation and the maximum difference between any two points is 9.46%. Figure 4.10 is the mean difference between the actual queue size and the foretold value of the queue size with a 95% confidence interval. From the data set of the entire experiment, the range is from -10 to 98. Each time slice also reveals the mean difference at that particular second.

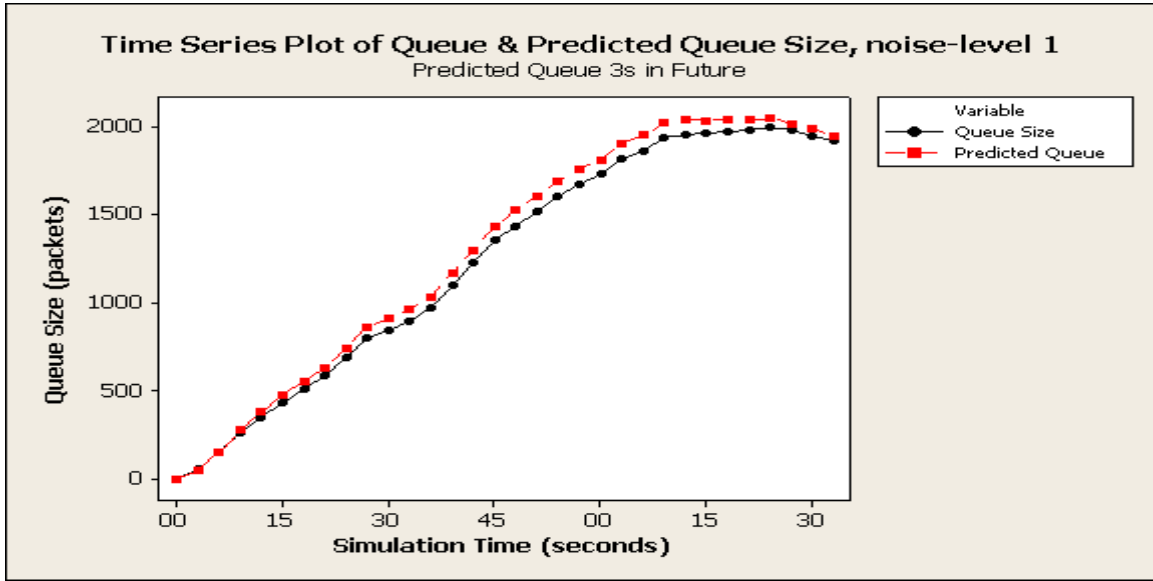


Figure 4.8: Compared Queue Size with Cholesky Factorization 1  
Given an input value of 1 to the Cholesky Factorization consideration for noise, the result of the predicted queue size versus the actual queue size vary only slightly.

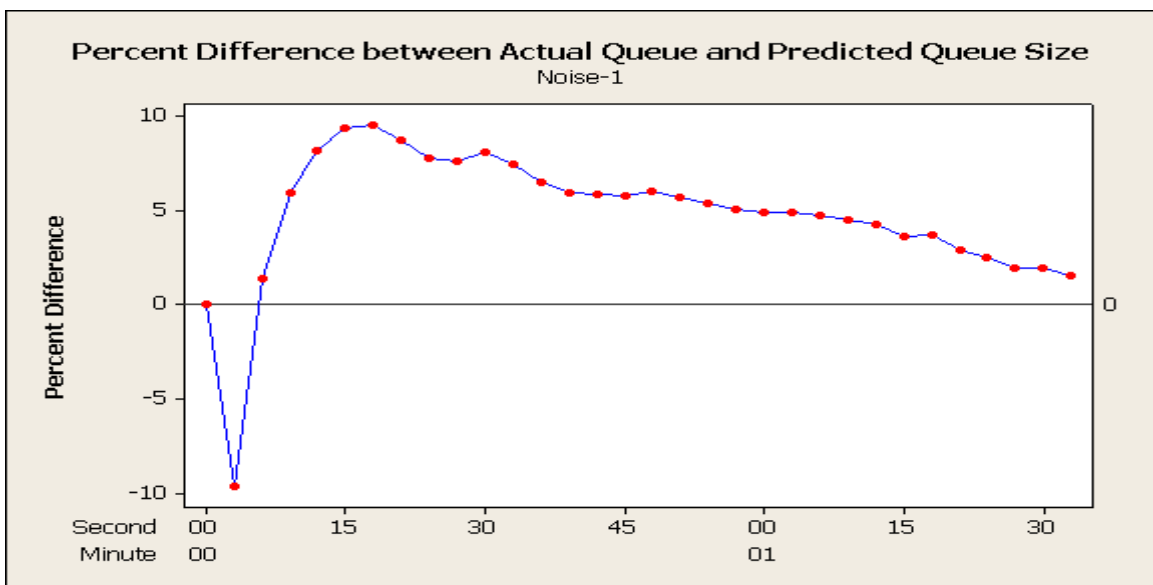
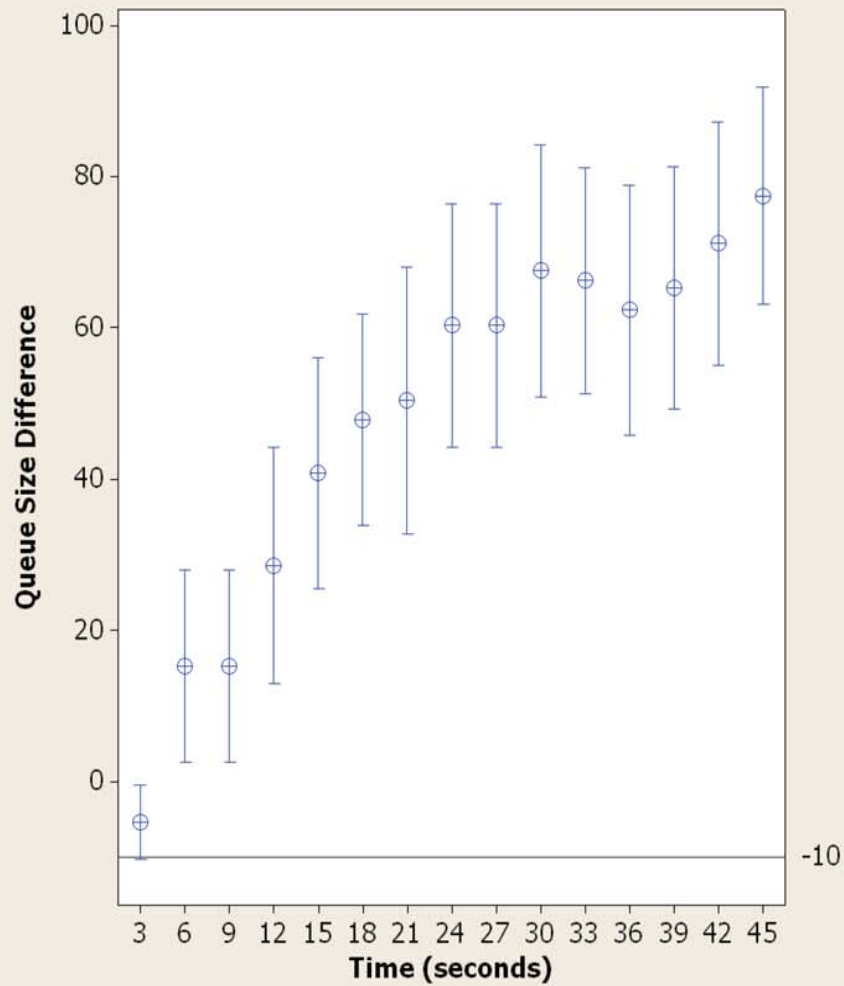


Figure 4.9: Percent Difference between Actual Queue Size and Predicted Queue Size  
During simulation the difference between the two values are calculated. The highest differing value is by 9.46% The values start to differ in the 5% range and below starting around 54 seconds.



**Actual vs Predicted Queue Size Difference (noise-1)**

95% CI for the Mean

**Actual vs Predicted Queue Size Difference (noise-1)**

95% CI for the Mean

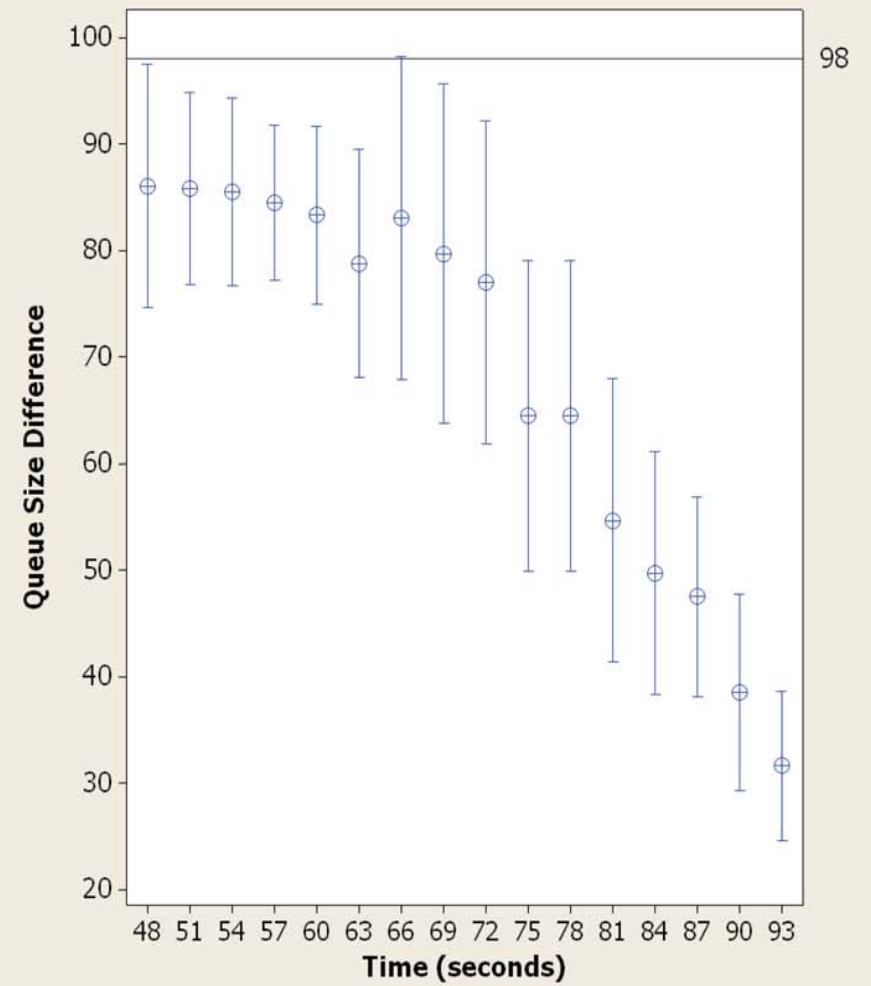


Figure 4.10: 95% Confidence Interval Mean Difference in Queue Size

95% Confidence Interval for the mean value of the difference between the actual queue size and predicted queue size. The noise parameter is set to 1 simulating a average level of noise in a network.

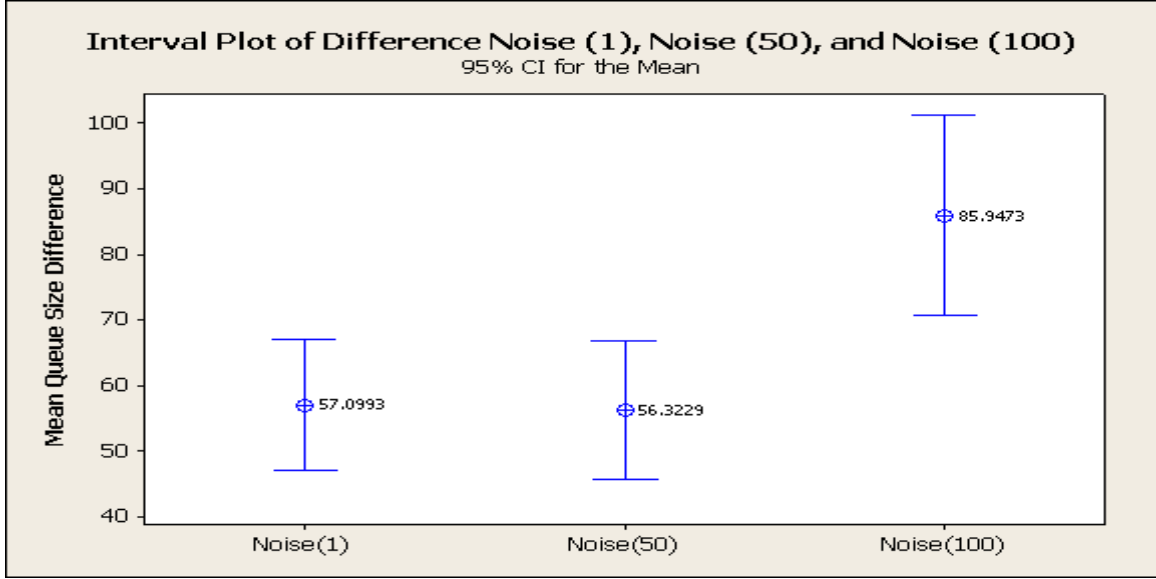


Figure 4.11: Interval Plot of Percent Differences with various noise parameters  
Comparison between various noise parameters of the precision of measured queue size versus future predicted queue size. Low level noise and moderate level noise can be tolerated and are statistically similar.

*4.1.1.4 Summary.* In summary, these experiments show that the noise parameters can be adjusted to simulate real-world network environment in order to get a range of noise that would cause the predicted values to become less accurate than can be tolerated. Figure 4.12 is all three experiments combined to see the comparisons. This indicated that the accuracy of the predicted value depends on the condition of the network noise level. The lower the noise the better the prediction of the network queue size. When normal level of noise Noise-1 and mid-range noise Noise-50, the prediction varied less than 5%, however when a large amount of noise was tested, the prediction varied by 8%. Figure 4.11 shows an interval plot of all three tested values. Noise-1 and Noise-50 are statistically equivalent, Noise-100 is not. This leads to the ability of the Kalman Filter to tolerate a somewhat noisy environment and it can still perform with error rate in the range of 5% from the actual queue size.

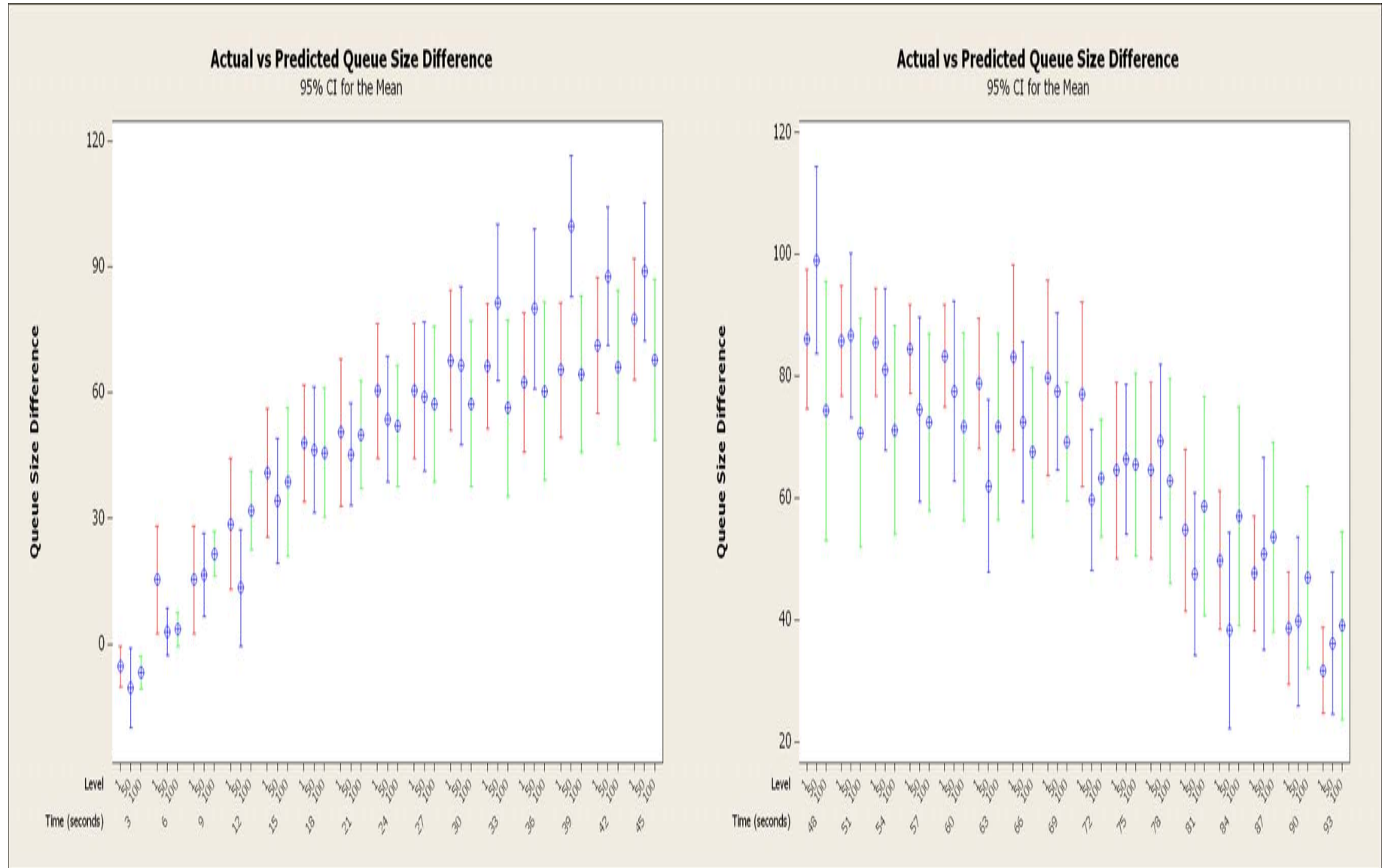


Figure 4.12: 95% Confidence Interval Mean Difference in Queue Size  
 95% Confidence Interval for the mean value of the difference between the actual queue size and predicted queue size for all three experiments

*4.1.2 How Far in the Future are Predictions Precise?* The next set of experiments involve how far in the future can the queue size be predicted with reasonable amount of accuracy. Again the network layout is depicted in Figure 4.1 and scenario with the same data links and traffic patterns are used as in above section.

*4.1.2.1 Experiment 1: Predict 5 Seconds in Future.* The Kalman filter was adjusted to predict five seconds into the future. Since the mean values are considered for the total runs of the simulation, some variability of the queue is not captured. Figure 4.13 is designed to show that if there are fluctuations in the queue size that the filter does capture those fluctuations. The figure is compiled from multiple runs of the scenario showing the queue size is raising and falling within the simulated tests.

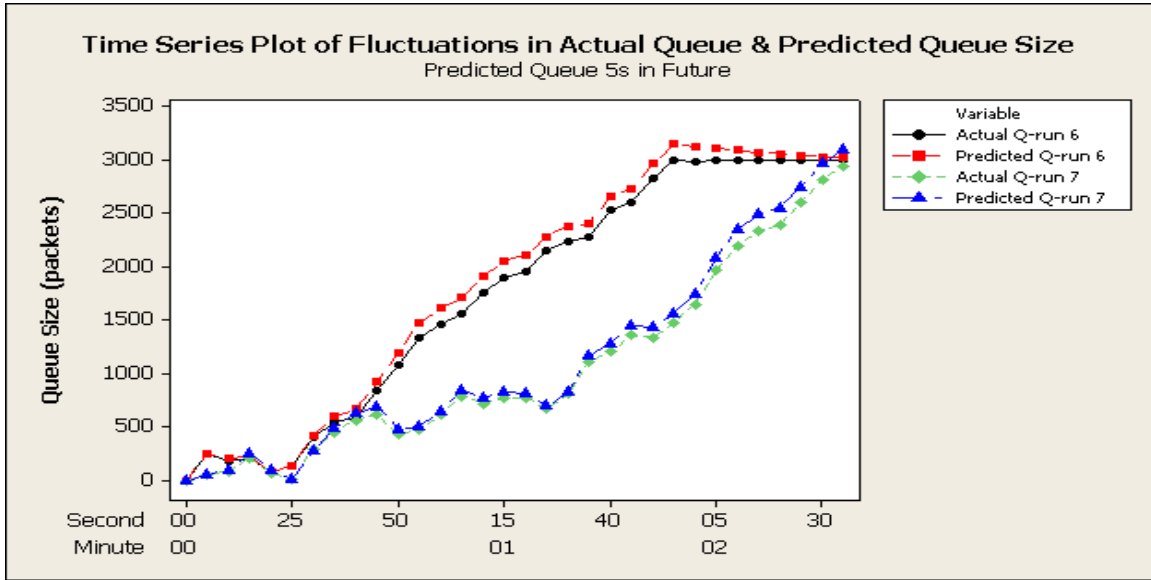


Figure 4.13: Time Series Plot of fluctuations in Queue Size  
Diagram showing the capability of the prediction module to capture the fluctuations of the actual queue size.

Figure 4.14 verifies that the forecasting is following the same trend as the actual queue size. The forecasted value stays relatively above the actual value. The mean value of the difference between the actual and predicted value is 6.66% with minimum difference, where the predicted value was below the actual size, was -.32% and the

maximum difference of 16.61%. Figure 4.15 is the representation of the percent differences. The overall summary of the results from this simulation is in the following graphs. Figure 4.16 shows the histogram of the size of the queues from the actual and predictions. It further indicates when the histogram are overlaid, where these differences are. The mean value of the queue size measured at 1876 with standard deviation of 1027, the mean value of the prediction measured at 1969 with standard deviation of 1037, giving a difference in measurement of 4.96%. Data was collected using a one-sample t-test with a 95% confidence interval. The interval is in the range of 4.866% to 8.459% with the standard deviation of 4.983.

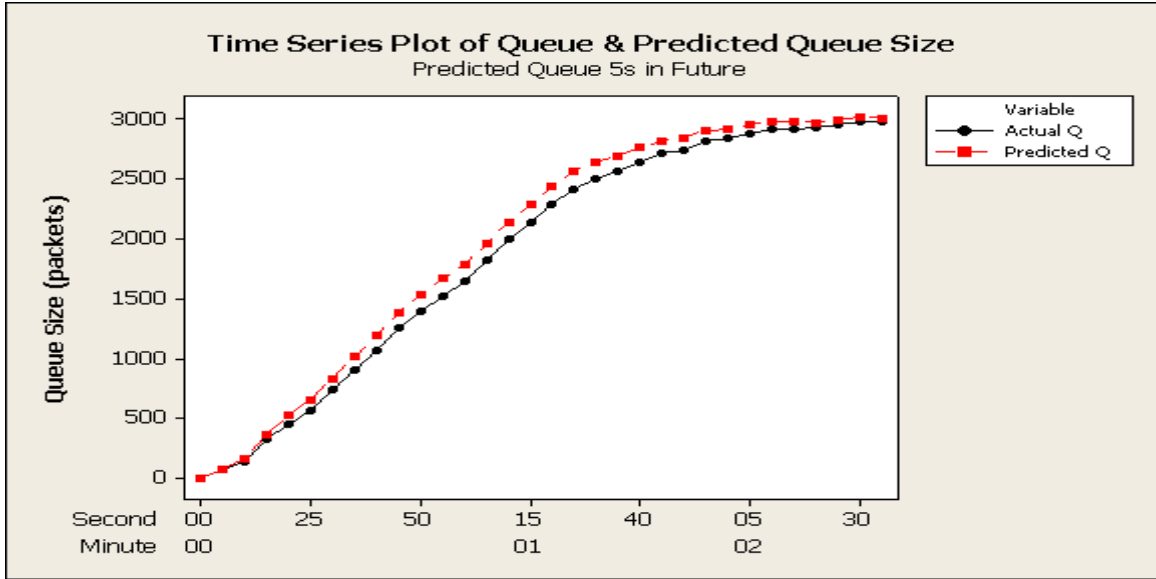


Figure 4.14: Time Series Plot of Actual Queue and 5 second Predicted Queue  
Diagram showing the actual queue size and predicted value of queue size taken 5 seconds in the future.

*4.1.2.2 Experiment 2: Predict 3 Seconds in Future.* Experiment two performed simulations with the Kalman filter set to forecast the queue size at three seconds in the future. The same network topology and scenario is being used as in Section 4.1. Figure 4.18 again demonstrates that the prediction follows the actual movement of packets to and from the queue in the network. Figure 4.19 reveals the mean percent difference of 4.81% with a forecasted measurement below the actual measurement of -10.30%. The maximum difference between the predicted queue size

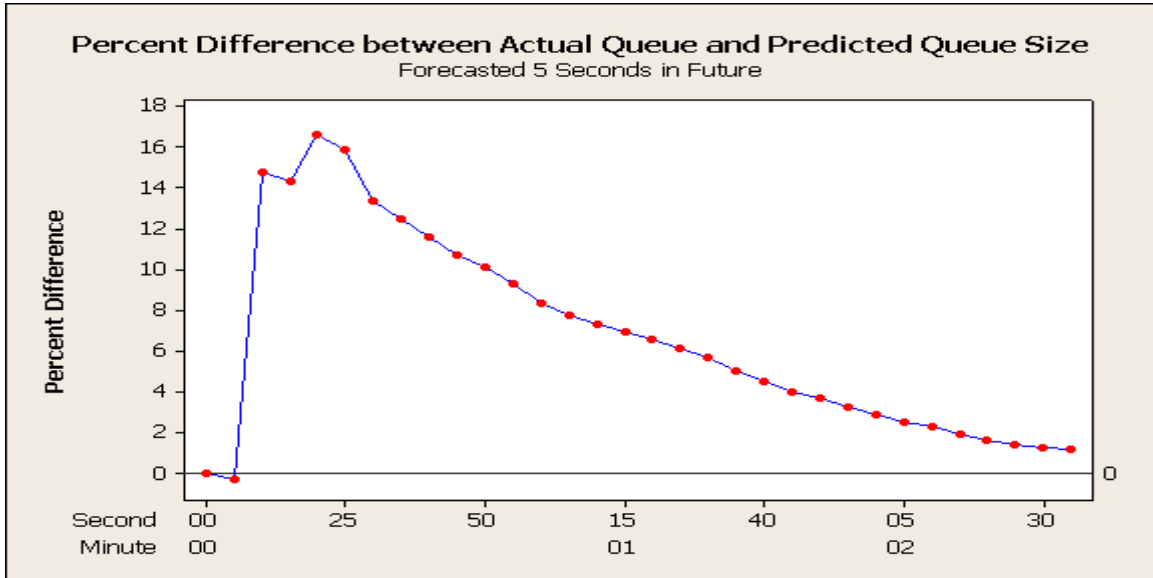


Figure 4.15: Percent Difference between Queue and 5 second Predicted Queue

The Percent Difference between the actual queue size and predicted value of queue size taken 5 seconds in the future yields a mean difference value of 6.66% with maximum difference of 16.61% and minimum difference of .32% below actual value.

and the actual monitored queue size is 9.65% with standard deviation of 3.58. The statistical results were gathered from using a one-sample t-test with a 95% confidence interval of 3.516% to 6.101%.

An overall histogram graph is shown to show the similarities and differences between the measurements, Figure 4.20. Also shown in Figure 4.21 is the 95% confidence interval of the entire run of the experiment showing the difference between the actual queue size and the predicted queue size.

*4.1.2.3 Experiment 3: Predict 1 Seconds in Future.* The last level of testing involves setting the prediction time to every second. This should yield more precise results compared to the other levels of testing. Figure 4.22 is given to show sample data where the actual queue size fluctuates during the simulation and that the predicted value continues to track with the actual queue size during those periods of fluctuations. Figure 4.23 shows the results of the simulation and verifies the precision of the tracking of the actual queue. The mean difference between the measurements is .56% with a standard deviation of 2.44. The minimum value difference is -8.38% being

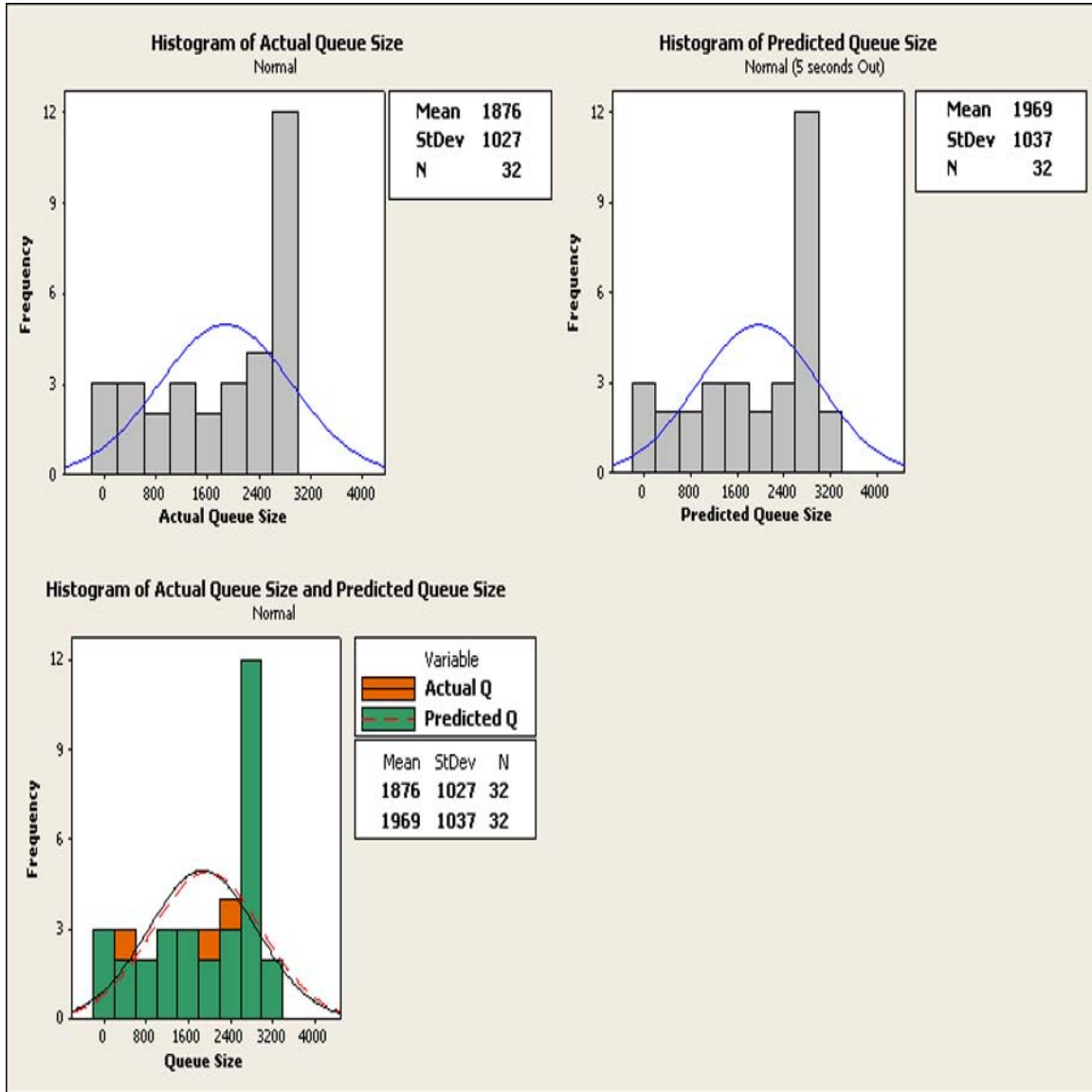


Figure 4.16: Histogram with normal distribution of queue sizes  
 Histogram with normal distribution of queue sizes. Actual queue size indicates the actual packets within the queue. Predicted queues size is the prediction of what the actual queue will be 5 seconds in the future. The overlaid histogram of both the actual and predicted values.

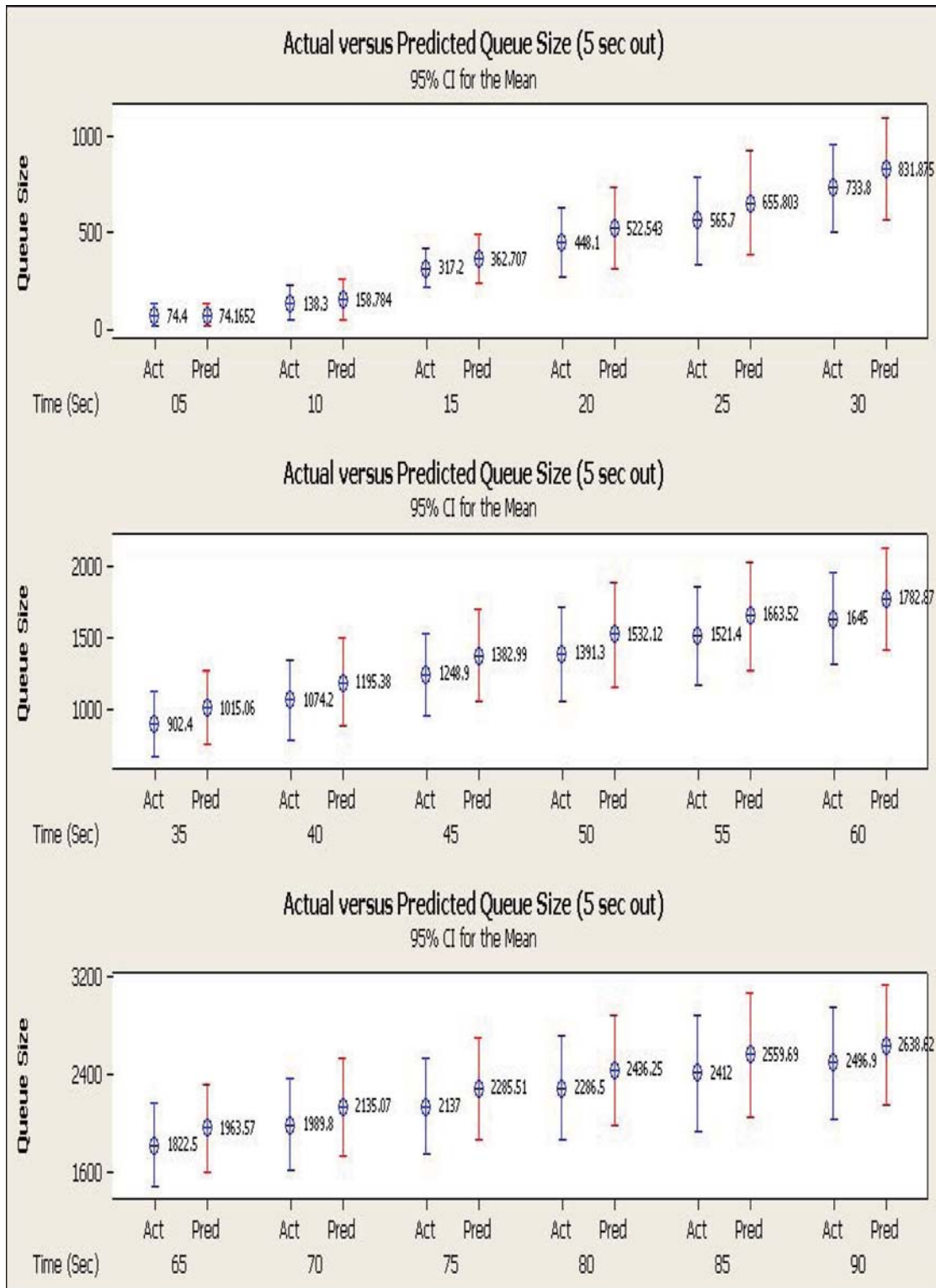


Figure 4.17: Total run of Actual vs Predicted Queue Size (5 seconds out)  
Interval chart of complete experiment predicting 5 seconds out for 155 seconds. Confidence interval at 95% for the mean of the actual and predicted queue size values.



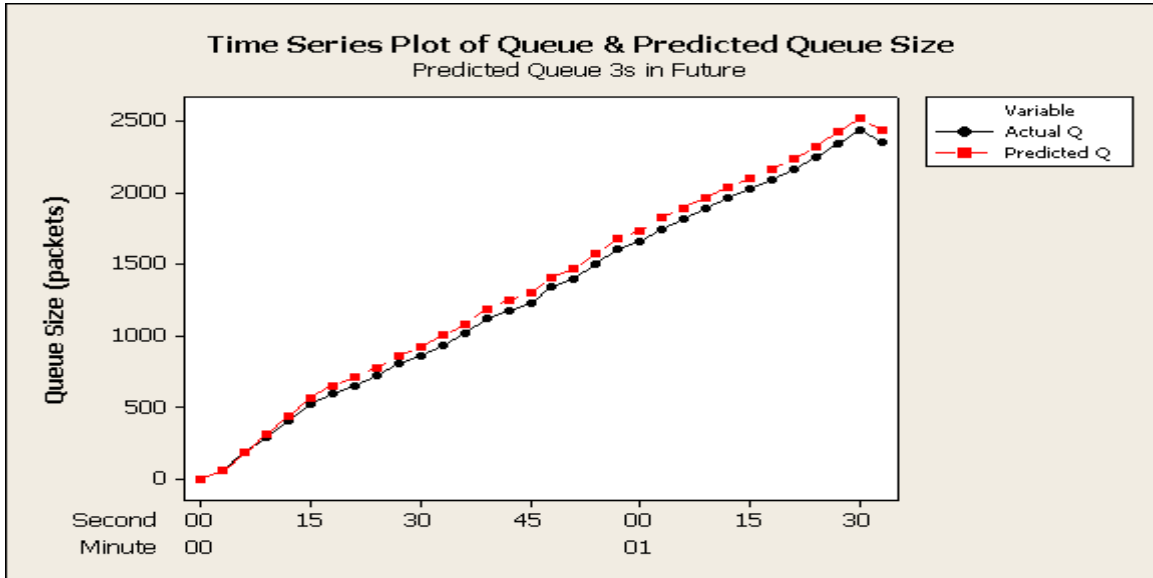


Figure 4.18: Time Series Plot of Actual Queue and 3 second Predicted Queue  
Diagram showing the actual queue size and predicted value of queue size taken 3 seconds in the future.

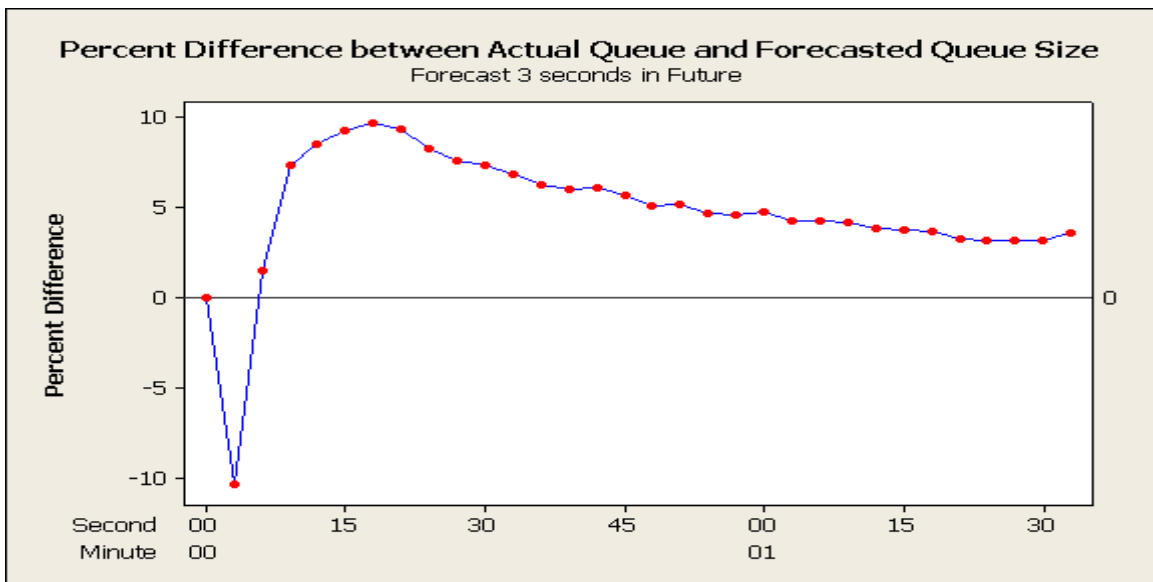


Figure 4.19: Percent Difference between the Actual Queue and 3 second Predicted Queue  
The Percent Difference between the actual queue size and predicted value of queue size taken 3 seconds in the future yields a mean difference value of 4.81% with maximum difference of 9.65% and minimum difference of -10.30% below actual value.

below the actual queue size and 2.27% overshooting the actual queue value. There was an outlier figure that I did discard, it was at the beginning of the simulation at time step 1 of -259% difference. Again, a one-sample t-test was used with a 95%

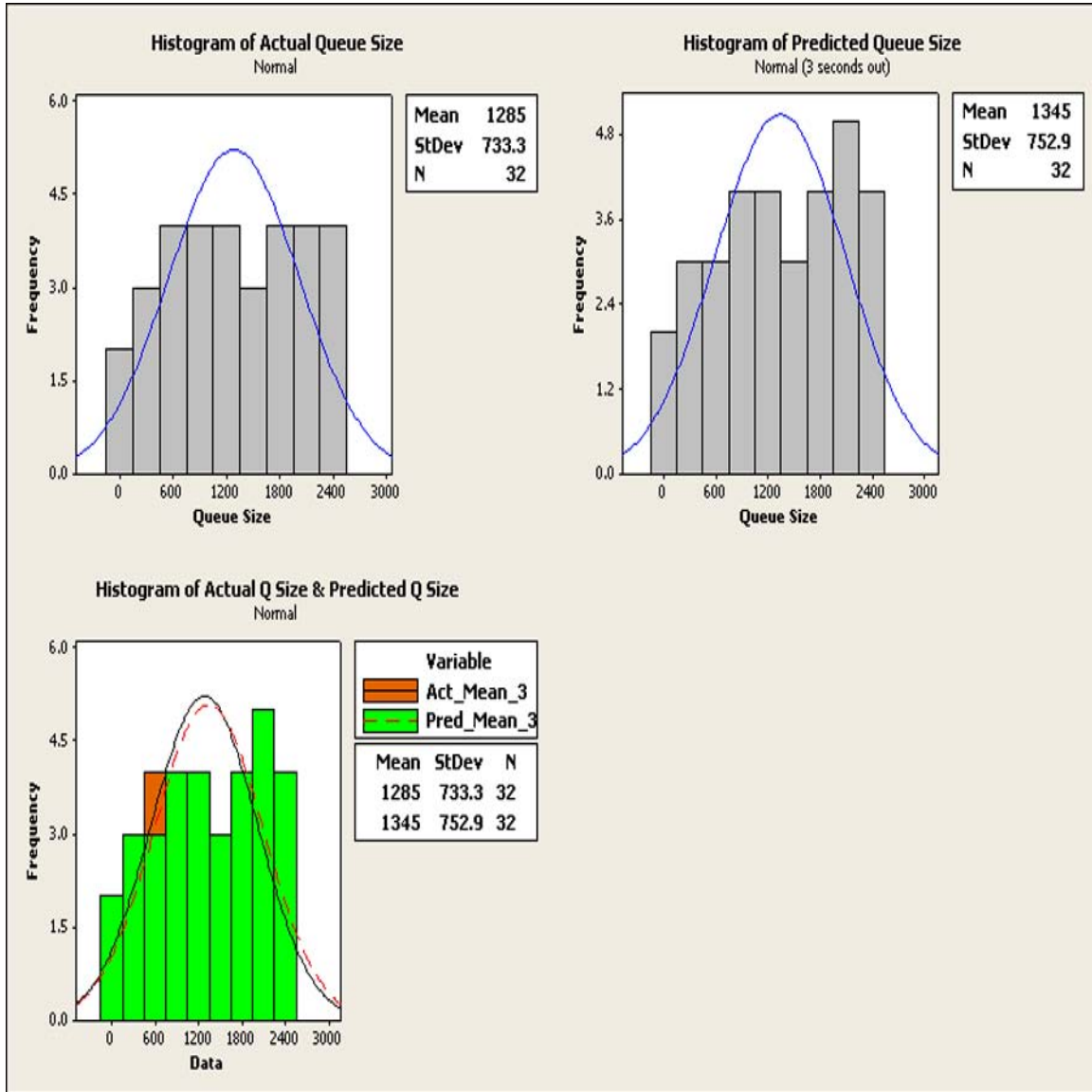


Figure 4.20: Histogram with normal distribution of queue sizes  
Histogram with normal distribution of queue sizes. Actual queue size indicates the actual packets within the queue. Predicted queues size is the prediction of what the actual queue will be 3 seconds in the future. The overlaid histogram of both the actual and predicted values.

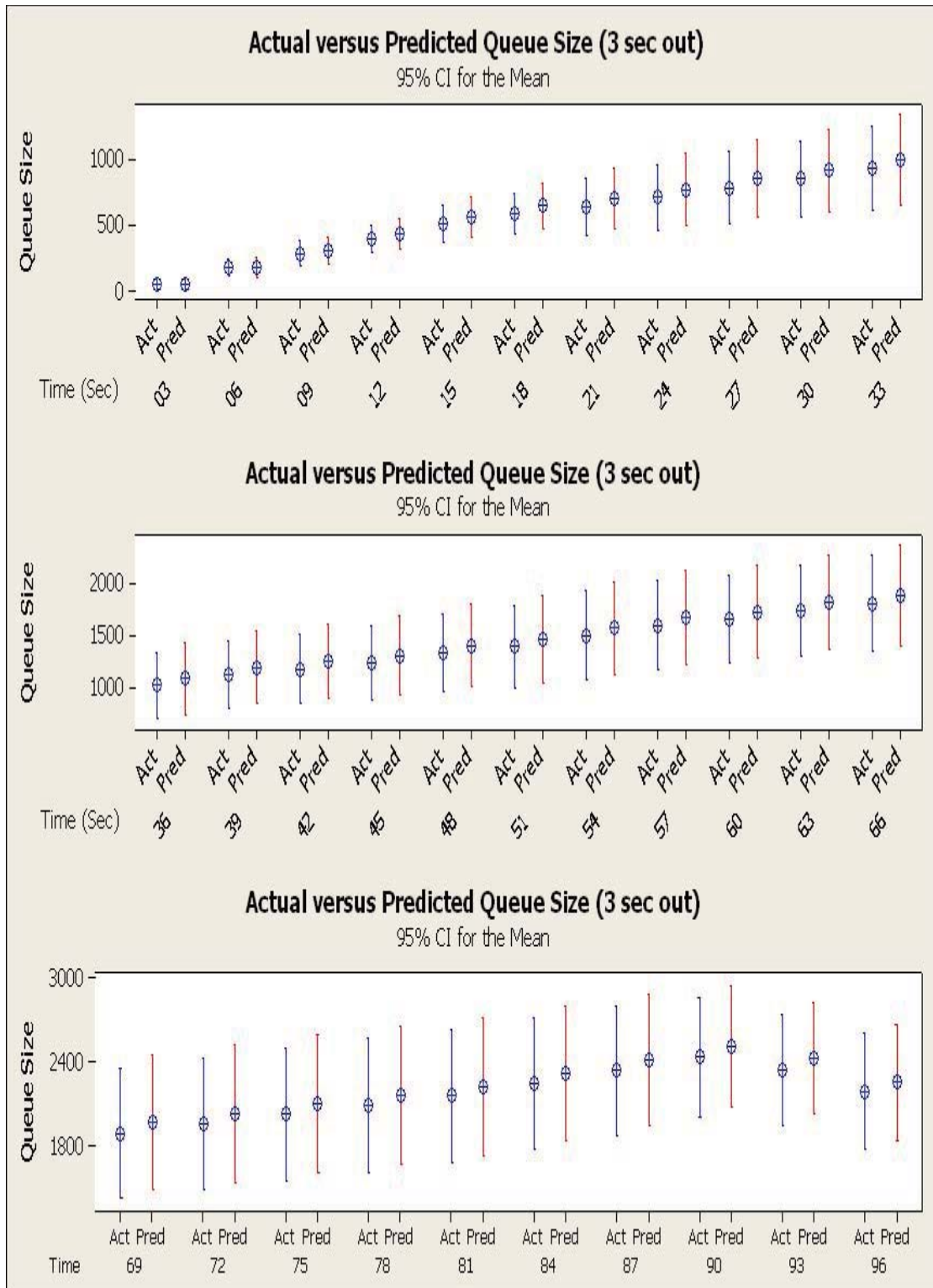


Figure 4.21: Total run of Actual vs Predicted Queue Size (3 seconds out)  
Interval chart of complete experiment predicting 3 seconds out for 96 seconds. Confidence interval at 95% for the mean of the actual and predicted queue size values.

confidence interval of  $-.325$  to  $1.467$ . Figure 4.25 and Figure 4.26 indicates at a 95% confidence interval at each second during the experiment what the actual queue size is and the predicted queue size.

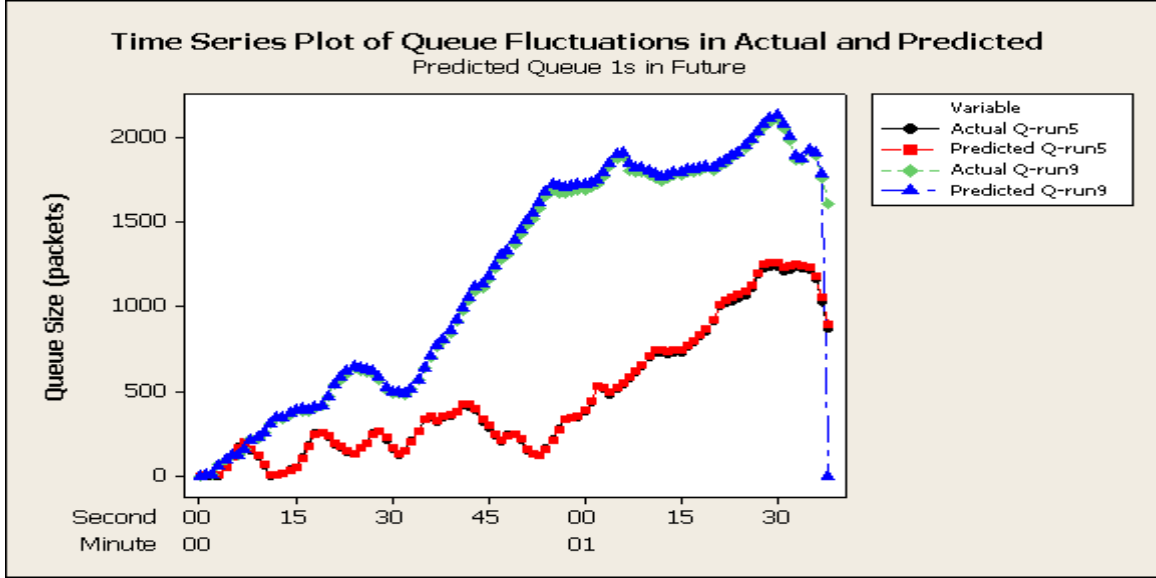


Figure 4.22: Time Series Plot of fluctuations in Queue Size

Diagram showing the capability of the prediction module to capture the fluctuations of the actual queue size.

*4.1.2.4 Summary.* In summary to the results from the simulation, the farther the outcast prediction the less accurate the prediction will be. The results start to vary by a maximum of 16.61% when forecasting out five seconds and a minimum of .57% when forecasting out 1 second in the future. Statistical analysis was accomplished using one-sample t-test and interval plots with a confidence interval of 95%. The interval plot in Figure 4.27 shows that the further out predictions are forecasted, the accuracy of the predictions decreases. The results also indicates that the predictions follows the same trends and fluctuations as the packets of information arrive and depart from the queue.

Based upon these experiments in the Network Prediction Module, the Hybrid Agent Network Control Module has been implemented with this feature. A circular queue is used to hold the predicted values, currently the size of this queue is five. An

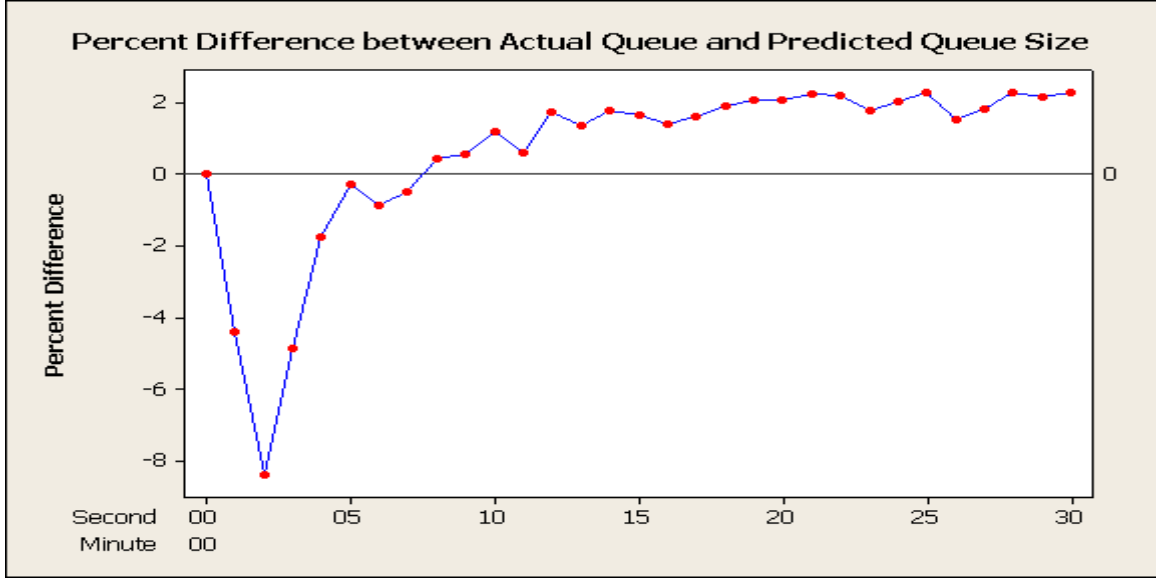


Figure 4.23: Percent Difference between the Queue Size and 1 second Predicted Queue  
The Percent Difference between the actual queue size and predicted value of queue size taken 1 seconds in the future yields a mean difference value of .57% with maximum difference of 2.27% and minimum difference of -8.38% below actual value.

average queue size is taken and based upon the change in the queue size, a current value and past value, a notification is sent to the sending node to throttle back or throttle forward message traffic. This takes a more proactive and decisive approach based upon future values and not reactive based upon actual congestion.

## 4.2 Decisive Routing and Admission Control Integration

This sections describes the integration of the three distinct frameworks and the advantages gained by the integration. The encryption optimization was only modified slightly due to time constraints, however, it is now working code and can be a future investigation. HANC was integrated with the ability to proactively advert congestion based upon the sensory readings from the network prediction module.

*4.2.1 Experiment 1: Encryption Optimization.* The modifications to this area were just adjustments to the compatibility with HANC and an extension of the EncryptFitter capability. Most of the original code was hardcoded into the HANC's

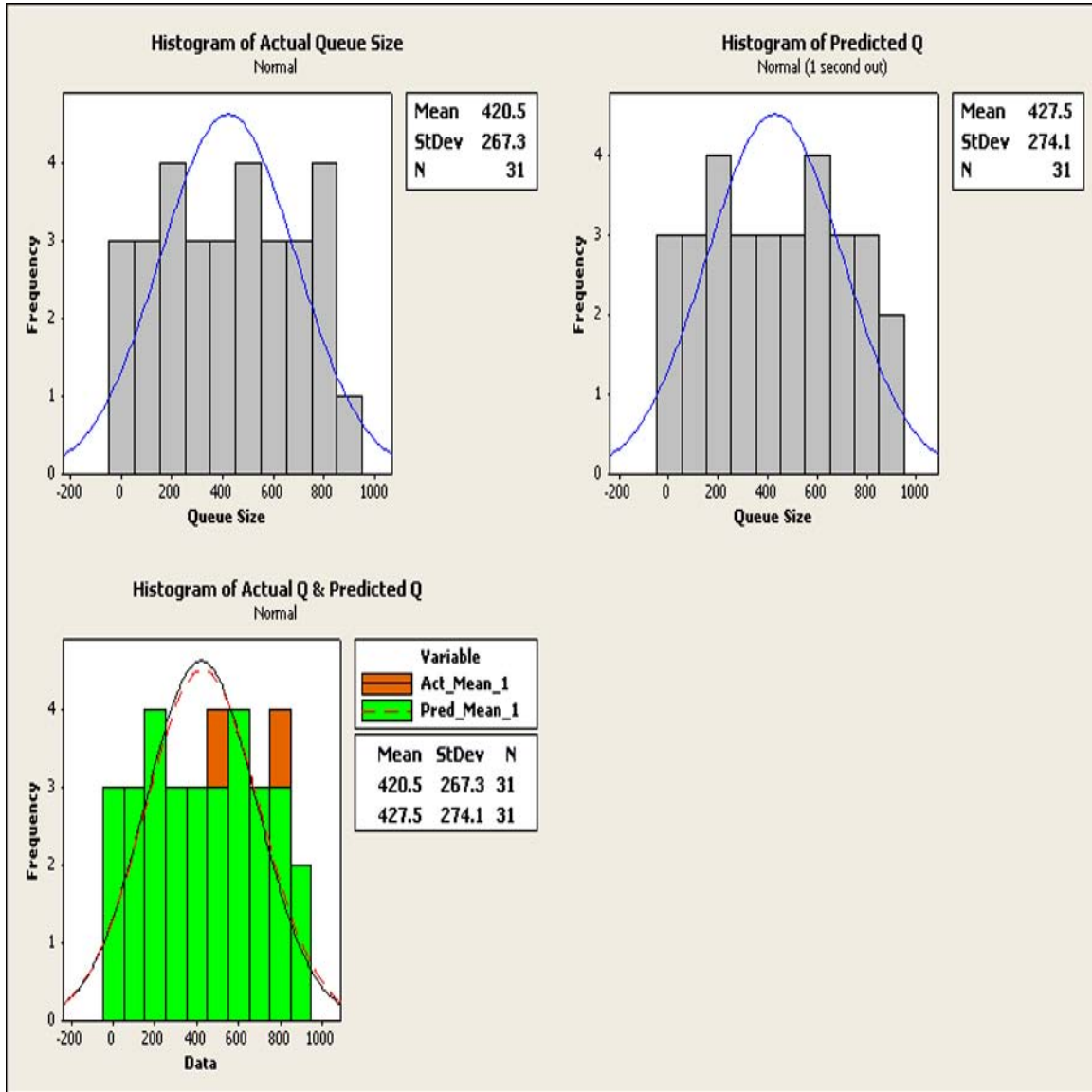


Figure 4.24: Histogram with normal distribution of queue sizes  
Histogram with normal distribution of queue sizes. Actual queue size indicates the actual packets within the queue. Predicted queues size is the prediction of what the actual queue will be 3 seconds in the future. The overlaid histogram of both the actual and predicted values.

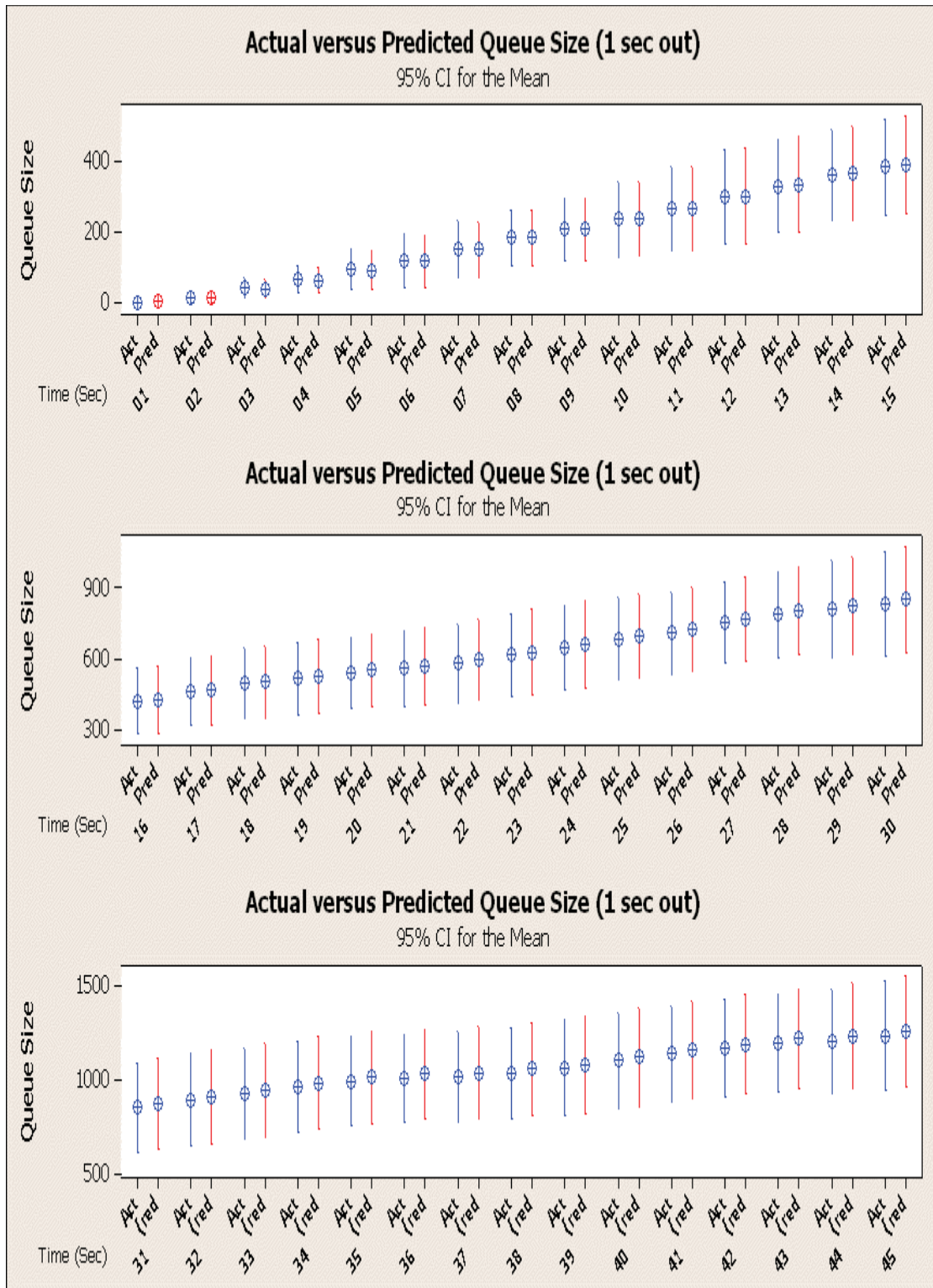


Figure 4.25: Total run of Actual vs Predicted Queue Size (3 seconds out)  
Interval chart of complete experiment predicting 1 seconds out for 90 seconds. Confidence interval at 95% for the mean of the actual and predicted queue size values.



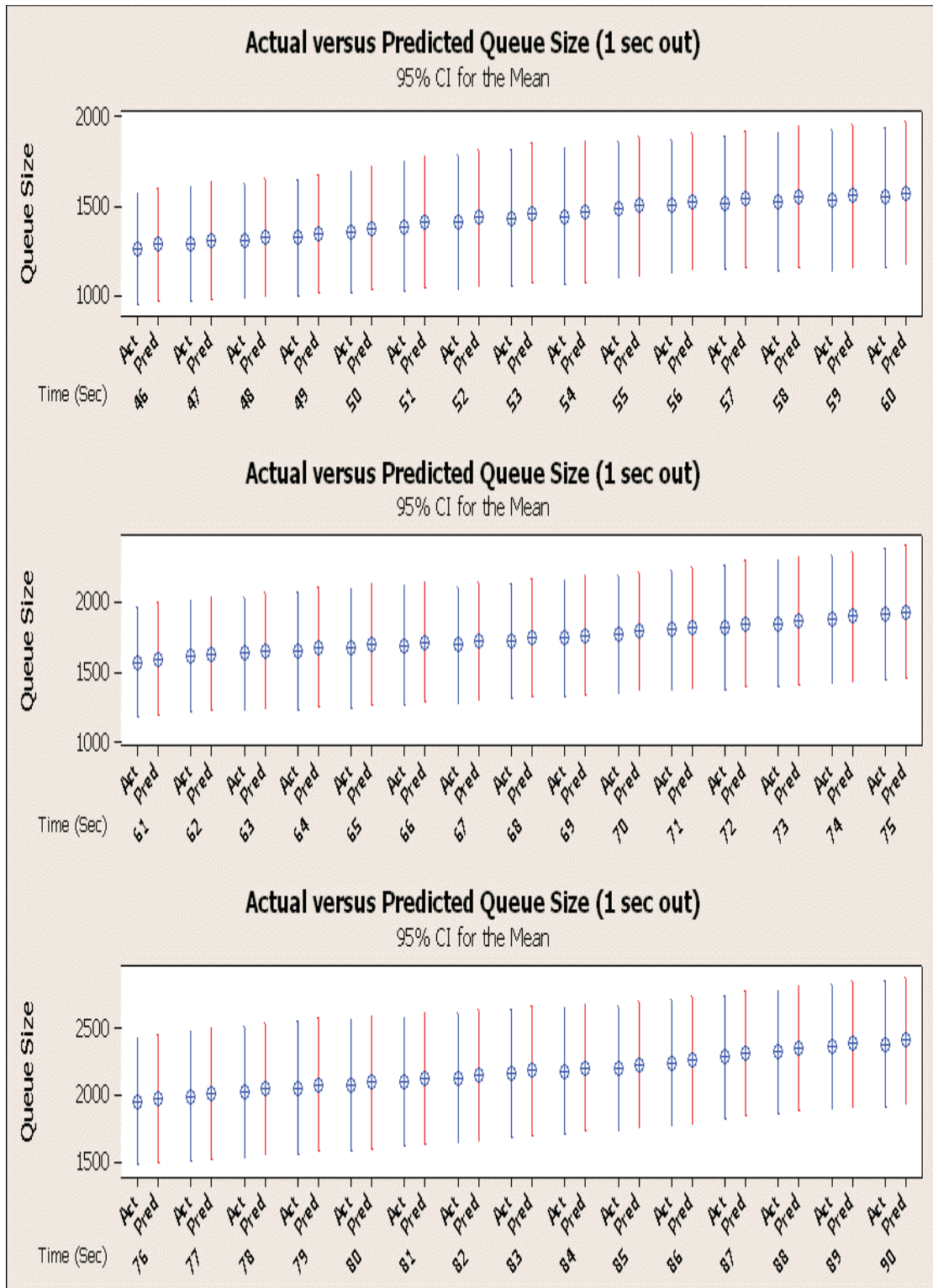


Figure 4.26: Total run of Actual vs Predicted Queue Size (3 seconds out)  
Interval chart of complete experiment predicting 1 seconds out for 90 seconds. Confidence interval at 95% for the mean of the actual and predicted queue size values.



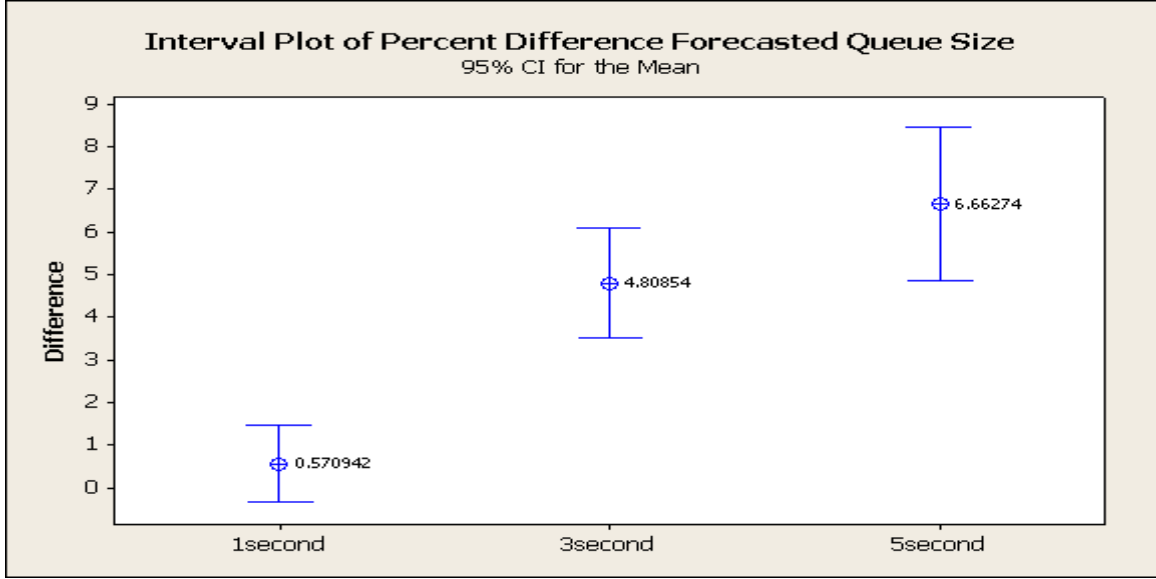


Figure 4.27: Percent Differences from Predictions at 1 sec, 3 sec, and 5 sec in future  
Diagram showing as predictions are forecasting further into the future the accuracy of the predictions decreases.

code, however, a **Matlab**<sup>®</sup> [9] file can be read in as the input parameters for the encryption optimizer. A field has been added to this input file to indicate the destination of each commodity. The input file must contain the bandwidth, CPU, commodity number, security level, performance level, and destination. An explanation of the security level and performance level can be found in Table 3.2. The method was tested to run one time during the simulation, an example of the input file that was tested is in Table 4.1. The output from the run of the optimizer is listed in Table 4.2, commodities three, four, and six were chosen as the optimal solution. Another set of inputs were given in Table 4.3 and the results in Table 4.4. Another enhancement was the calling method was adjusted to receive an input parameter of the requesting node, therefore HANC knows what node to send the commodity from and what node is the receiving node.

*4.2.2 Experiment 2: Preemptive Congestion Control.* This level examines the use of the prediction to preemptively lower congestion based upon the forecasted network state. The topology used is in Figure 4.28. There are seven nodes with

Table 4.1: Encryption Optimization Input Parameters 1

Available Bandwidth (MB)	Available CPU (sec)	-	-	-	-
100	2400	0	0	0	0
Commodity Number	Size	Priority	Security Level	Performance Level	Destination
1	50	60	5	1	7
2	25	28	5	2	7
3	30	90	5	3	7
4	5	30	5	1	7
5	10	15	4	2	7
6	10	75	1	1	7
7	100	90	5	3	7
8	22	49	2	2	7
9	20	74	2	3	7
10	44	38	5	2	7

Table 4.2: Encryption Optimization Output 1

ID	In Size	Prior-ity	Sec Level	Perf Level	Dest	Rand-om	Algo	Comp-ress	Key Size	Out Size	Encrypt Time
3	30	90	5	3	7	0	0	0	256	30.000097	1416
4	5	30	5	1	7	0	5	0	4096	5.001123	413.742
6	10	75	1	1	7	0	6	0	1536	10.00288	519.563

Table 4.3: Encryption Optimization Input Parameters 2

Available Bandwidth (MB)	Available CPU (sec)	-	-	-	-
100	2400	0	0	0	0
Commodity Number	Size	Priority	Security Level	Performance Level	Destination
1	80	60	3	1	3
2	15	28	3	2	3
3	30	90	3	3	3
4	5	30	2	1	3
5	50	15	4	2	3
6	60	75	1	1	3
7	100	90	5	3	3
8	22	49	2	2	3
9	20	74	2	3	3
10	44	38	5	2	3

Table 4.4: Encryption Optimization Output 2

ID	In Size	Prior-ity	Sec Level	Perf Level	Dest	Rand-om	Algo	Comp-ress	Key Size	Out Size	Encrypt Time
3	30	90	3	3	3	0	0	0	128	30.000097	1212.061
9	20	74	2	3	3	0	3	0	128	10.000065	1039.338

duplex links, the Kalman filter resides at the outbound side of Node 3 to Node 7. The link between Nodes 3 and 7 are two simplex links to capture the filter data on the outbound side only. The links are 4Mb 8ms drop tail, except the link between Nodes 2 and Node 3, and back as well as Node 3 to 7. These links are 1.5Mb capacity with 8ms delay. HANC produces its own traffic so a traffic generator was not used. It is sort of a looping affect, once a node receives traffic, it starts to send traffic. The simulation is timed to run for three seconds. The queue limit at the Kalman filter node is set to 1000. The Kalman filter timer is set to gather readings at every 0.2 seconds

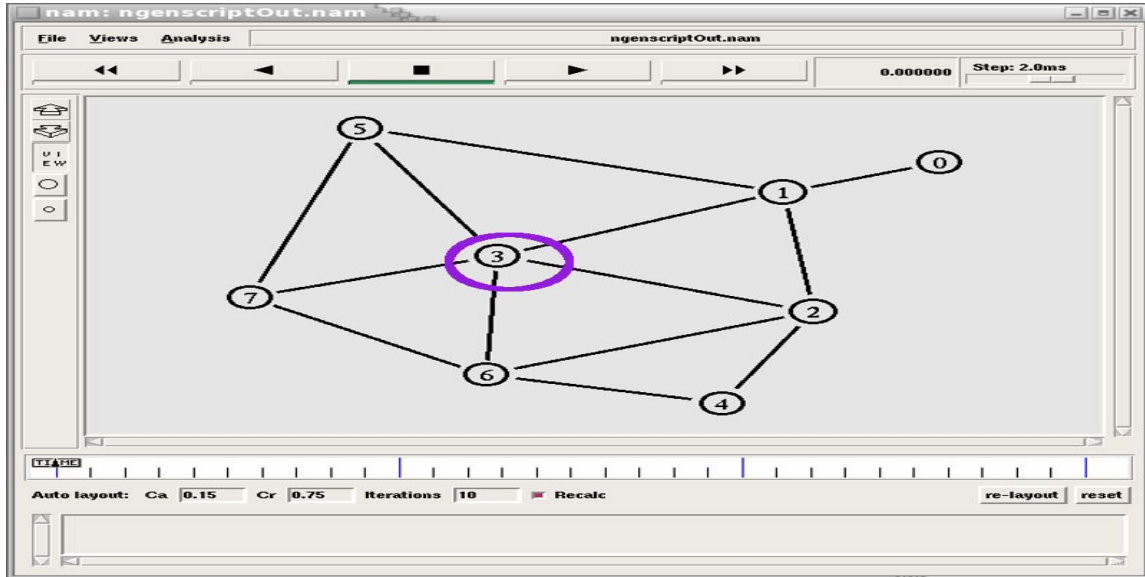


Figure 4.28: Network Topology for Integration Testing

Network Topology consists of 8 nodes. The Kalman filter is at Node 3's outbound link to Node 7.

A delta parameter is being calculated by taking the forecasted queue size readings from the Kalman filter at Node 3 and storing the values in a circular queue data structure. The queue is currently set to hold five values, as data comes in the next

value will be overwritten in a circular fashion. An average is taken of the readings in the queue and stored in variable current delta, it will take up to five readings in order for this value be of good use. As new values come in, the last current delta is switched and stored in a variable previous delta and the current delta is the replaced with the updated value.

The difference between the current delta and previous delta are compared, if the current delta is larger, this indicates that the queue is filling up, storage capacity is decreasing. If the current delta is smaller than the previous, the queue storage capacity is becoming larger. Figure 4.29 shows traffic going through the network and the queue is starting to fill up at Node 3.

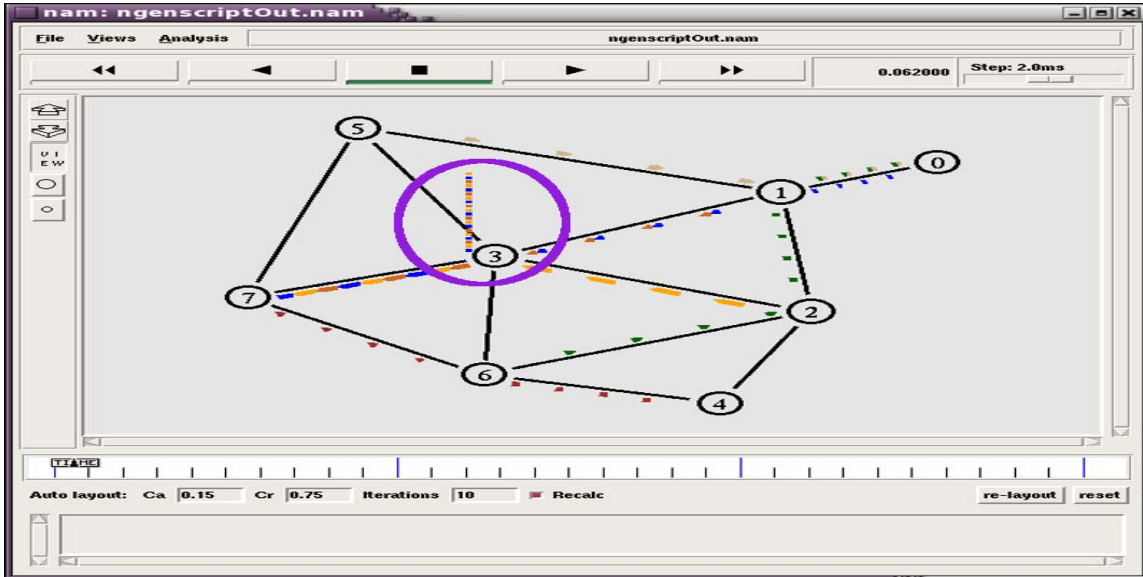
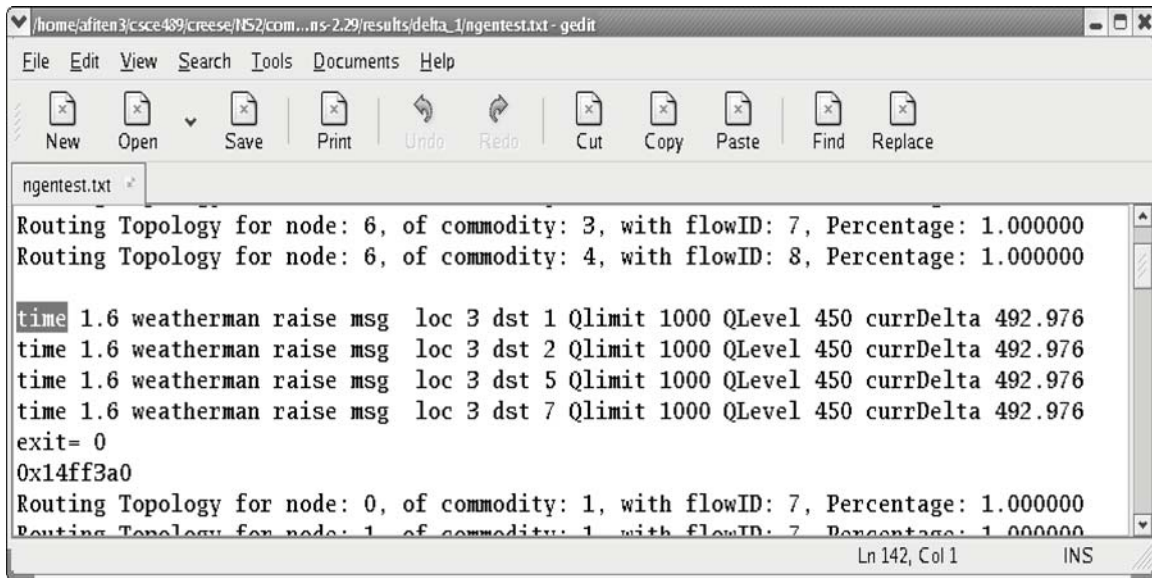


Figure 4.29: Network Topology for Integration Testing, Kalman Filter queue storing data Network Topology consists of 8 nodes. The Kalman filter is at Node 3's outbound link to Node 7. The queue is starting to fill up with packets being received.

A decision to notify a node's neighbors is based upon the queue storage capacity decreasing to 45% of its total capacity. This is an arbitrary number that I have chosen, but it could be a customer choice of what's tolerable. If the level is reached, the preemptive throttle back messages goes out to the neighbors giving time to respond with either stopping traffic, or sending only the most important of its traffic. Figure 4.30 show that in the code, the message is being sent at simulation time of

1.6 to all Node 3's neighboring nodes. Figure 4.31 shows "raise threshold" messages, circled in purple, being sent from Node 3 to its neighbors that correlates to code result snippet of Figure 4.30. Figure 4.32 also shows the red colored messages going out from Node 3 again at a different simulation time. The "raise threshold" messages are circled in purple again. Finally, the offending nodes takes action on the throttle back message received from Node 3 and this is shown in Figure 4.33. The action that Nodes 1 and 2 take is stopping message traffic being sent to Node 3. The figure also depicts that the queue has past the 45% and now is sending "raise threshold" messages more frequently, this is also due to the now reactive nature of HANC.



```

/home/afiten3/csce489/creese/NS2/com...ns-2.29/results/delta_1/ngentest.txt - gedit
File Edit View Search Tools Documents Help
New Open Save Print Undo Redo Cut Copy Paste Find Replace
ngentest.txt
Routing Topology for node: 6, of commodity: 3, with flowID: 7, Percentage: 1.000000
Routing Topology for node: 6, of commodity: 4, with flowID: 8, Percentage: 1.000000
time 1.6 weatherman raise msg loc 3 dst 1 Qlimit 1000 QLevel 450 currDelta 492.976
time 1.6 weatherman raise msg loc 3 dst 2 Qlimit 1000 QLevel 450 currDelta 492.976
time 1.6 weatherman raise msg loc 3 dst 5 Qlimit 1000 QLevel 450 currDelta 492.976
time 1.6 weatherman raise msg loc 3 dst 7 Qlimit 1000 QLevel 450 currDelta 492.976
exit= 0
0x14ff3a0
Routing Topology for node: 0, of commodity: 1, with flowID: 7, Percentage: 1.000000
Routing Topology for node: 1, of commodity: 1, with flowID: 7, Percentage: 1.000000
Ln 142, Col 1 INS

```

Figure 4.30: Preemptive Congestion Control Code Snippet

The Decisive Routing and Admission Control According to Quality of Service Constraints code snippet of reaction to forecasted state of the network. The Kalman filter queue has reached a stated level of 45% of its capacity and action is taken.

In order to visually see and distinguish the preemptive messages, I had to turn off all messages that have the same flow ID, such as the "Alive" messages. Trying to change the flow ID caused the program to crash.

*4.2.3 Summary.* In summary to these battery of tests, bringing together three distinct frameworks into one cohesive framework is advantageous to information technology. Tweaking the encryption optimization allows for more flexibility in

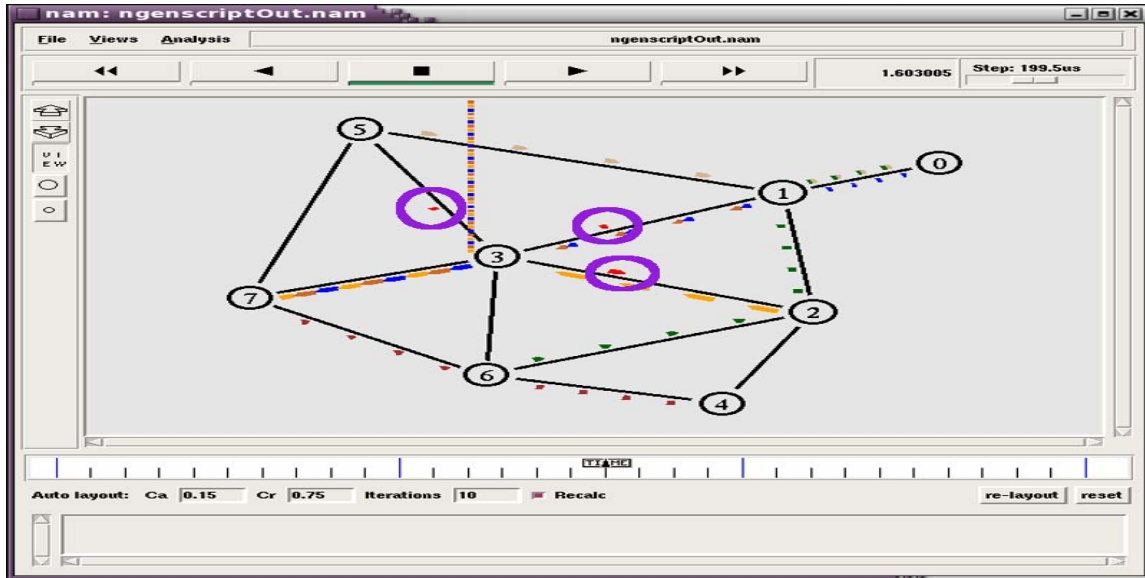


Figure 4.31: Preemptive Congestion Control Throttle Back Message First Occurance  
The Decisive Routing and Admission Control According to Quality of Service Constraints simulation snippet of reaction to forecasted state of the network. The Kalman filter queue has reached a stated level of 45% of its capacity and throttle back messages are being sent from Node 3.

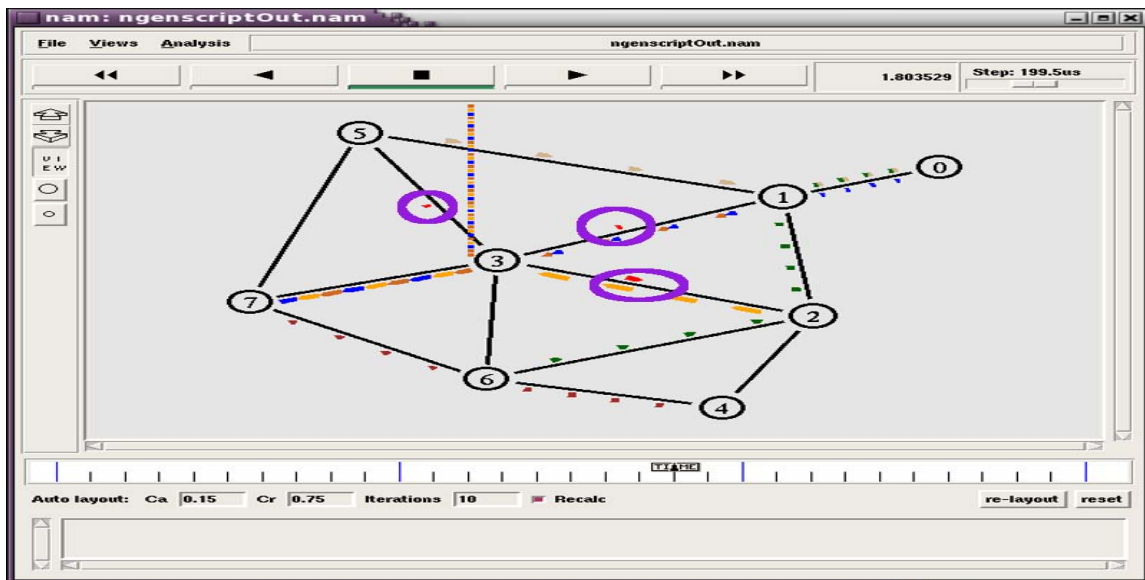


Figure 4.32: Preemptive Congestion Control Throttle Back Message Second Occurance  
The Decisive Routing and Admission Control According to Quality of Service Constraints simulation snippet of reaction to forecasted state of the network. The Kalman filter queue has reached a stated level of 45% of its capacity and throttle back messages are being sent from Node 3.

adding additional parameters as input. Adding the addition of a sending source and destination, proves to mimic a real world event where there are classified documents

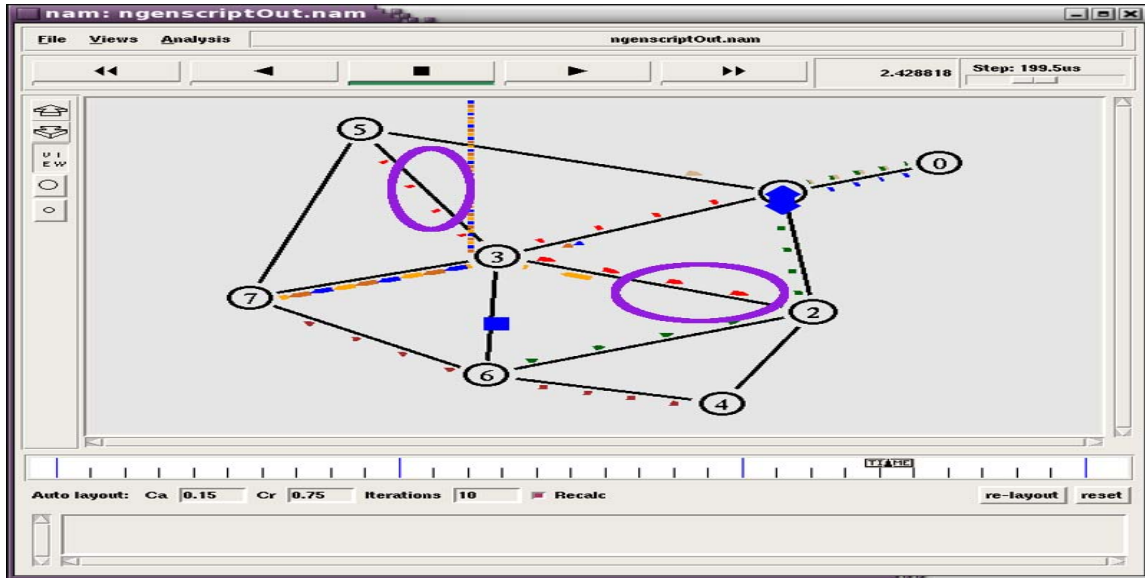


Figure 4.33: Preemptive Congestion Control Response to Throttle Back Messages  
 Preemptive Congestion Control Throttle Back Message cause offending nodes to stop sending traffic. However the queue did loose a few packets before traffic was completely shut off.

that may need to go to several locations, but there is the need to only use the network with an optimal solution due to the constraints of the network.

Combining the Network Prediction Module proves to provide proactive action when the state of the network is outside a tolerable range. When an additional threshold is reached, then HANC responds reactively as well to control the congestion.

## V. Conclusions

### 5.1 *Summary of Research*

To restate the motivation behind this research, information superiority is a key factor in military operations. According to Joint Pub 3-13, the ability to collect, process, and disseminate an uninterrupted flow of information while exploiting and/or denying an adversary's ability to do the same is the definition of information superiority [19]. It further states that to be successful, the conduct of operations requires access to information available outside the operational area and that warfighters need frequent, instant, and reliable access to information at locations in the continental United States as well as in theater. The aim was to provide a framework that would enhance the forward commander's ability to capture the needed information to make reliable decisions.

The framework was developed from the consolidation of separate and distinct tools to build upon the advantages and gains each tool brings. One distinct tool was the ability to forecast the state of a network queue. The advantage of this is that if a commander can have a sight picture of the network, they could make proactive and smarter decisions about the traffic flow and getting important data to the warfighters the optimal way possible. This tool uses stochastic estimation and control theory through the use of a Kalman filter that resides at a network node's level. This tool is likened to a Heating Ventilation and Air Conditioning system but on the scale of a computer network. Given the current state of the "room", parameters can be adjusted, sensory readings taken, to produce the "temperature" that is wanted. There is an integration of the drop tail queuing policy and the Kalman filter timed predictions, as a packet is received sensory data is taken to predict what the actual size of the queue will be sometime in the future. The closer in the predictions are, the lower the error rate of predictions. However, since one second out may be an unrealistic time frame to take any proactive action, the five seconds out test still gave only a 9.65% difference to the actual queue. If the state of the queue is known, congestion can be thwarted and alternative routes can be used. Out-range predictions



can also go along with preplanned schedules. If what is planned is changed or the network changes, what's in the plan could possibly be reworked to have an optimal or better use of resources.

Another tool is the ability to have document encryption optimization based upon the state of the network as well. As the network dynamically changes state, an encryption optimization module can produce an optimal path and list of documents that can be sent to the destination. Having a repository of encryption algorithms schemes, the ones integrated into this research area were RSA, ELGamal, AES, 3DES, TwoFish, BlowFish, and CAST5. As well as customer based security level and performance parameters, this tool uses glpk to optimize the best possible solution given bandwidth, CPU requirements, and file sizes. With this adds the benefits of utilizing a perceived state sometime in the future to develop the list of commodities as fluctuations in parameter. Predictions can be tuned which in turn can dynamically change the list of documents that must go out. Further, if there are last minute add-in/on's, the sight picture can show the window of opportunity to send such items.

The last tool that was integrated was the Hybrid Agent for Network Control. This tool represents the smart agent at the node level that would know how to process and handle the network weather picture and encryption optimizer as well have its own onboard congestion control scheme based upon priority level of data and current congestion levels. HANC is based upon a robot control architecture. The network has defined behaviors based upon the stimulus it receives. The new behaviors that have been adopted were the behavior of reacting to proactive predictions, if the queue gets within a certain range, an action occurs. Also the behavior of sending messages categorized as being encrypted messages.

## **5.2 *Future Research***

My overall goal was to integrate three separate frameworks capitalizing on the sum of the individual parts thereby creating a Decisive Routing and Admission Con-

trol according to Quality of Service Constraints. Tests show preliminary positive results and warrants possibly refactoring the code and enhancing the response actions of the nodes receiving the “raise threshold” messages. Future research and study in the direction of evaluating what additional potential the Kalman filter has given other queuing schemes. The Kalman filter calculations proved to become unstable using exponential traffic and very large link capacity and bytes of data, this area can be investigated.

Another area of research could be in testing further ranges of predictions, and fine tuning the capability and size and/or type of the data structures used, the simulations used a circular queue and capturing five data values. Less data values may increase the accuracy rate and the timing of neighboring nodes response. Testing was only done with a system with one Kalman filter drop tail design acting simply as a router, to capture the results. The system can handle multiple filters, which were also tested, however, time didn’t permit creation and capture of those data results.

The encryption module was just changed and modified to accept the destination of the encryption message as well as integration into HANC produce stable results. This begs further research because encryption optimization routing is vitally import. One could research HANC becoming the smart agent directing the optimization based upon bandwidth forecasted or realized in the network. A parameter could be incorporated based upon the sensory readings from the Network Prediction Module.

## Bibliography

1. Adams, Carlisle M. "Constructing Symmetric Ciphers Using CAST Design Procedure". *Designs, Codes, and Cryptography*, 12 Num 3, 1997.
2. Bolch, Gunter, Stefan Greiner, Hermann de Meer, and Kishor S. Trivedi. *Queueing Networks and Markov Chains: Modeling and Performance Evaluation with Computer Science Applications*. Wiley-Interscience, 2006.
3. Boorstyn, Robert and Adam Livne. "A Technique for Adaptive Routing in Networks". *Transactions on Communications*, volume 29 Num 4, 474–480. IEEE, 1981.
4. Chadda, Ankur. *QoS Testing Methodology*. Master's thesis, University of New Hampshire, December 2004.
5. Compton, M., K. Hopkinson, and S. Graham. "The Network Tasking Order (NTO)". *Proceedings of the IEEE Military Communication Conference*, 2008.
6. Federal Information Processing Standards Publication 197. *Advanced Encryption Standard (AES)*. National Institute of Standards and Technology (NIST), November 26, 2001.
7. GNUPg.org. URL [www.gnupg.org](http://www.gnupg.org).
8. Hintz, Kenneth J. "GMUGLE: A Goal Lattice Constructor". *SPIE*, 4380:324–327, April 2001.
9. Matlab. URL [www.mathworks.com](http://www.mathworks.com).
10. McIntyre, Gregory A. *A comprehensive Approach to Sensor Management and Scheduling*. Ph.D. thesis, George Mason University, 1998.
11. Menezes, Alfred J., Paul C. Van Oorschot, and Scott A. Vanstone. *Handbook of Applied Cryptography*. CRC-Press, 1 edition, 2001.
12. MyCrypto.net. URL [www.mycrypto.net](http://www.mycrypto.net).
13. Network Simulator Version 2. URL [www.isi.edu/nsnam/ns/](http://www.isi.edu/nsnam/ns/).
14. Pecarina, John M. *Creating An Agent Based Framework To Maximize Information Utility*. Master's thesis, Graduate School of Engineering, Air Force Institute of Technology (AETC), Wright-Patterson AFB OH, March 2008. AFIT/GCS/ENG/08-19.
15. Robertazzi, Thomas G. *Computer Networks and Systems Queueing Theory and Performance Evaluation*. Springer, 2000.
16. Schneier, Bruce, John Kelsey, Doug Whiting, and et al. *Two Fish: A 128-bit Block Cipher*. Wiley & Sons, June 15, 1998.

17. SECAF/CSAF. "Mission Statement and Priorities Letter to Airman", 15 September 2008. URL <http://www.af.mil/library/viewpoints/jvp.asp?id=401>.
18. Sheldon, Tom. *Encyclopedia of Networking*. McGraw-Hill Companies, electronic edition, 1998.
19. of Staff, Joint Chiefs. *Information Operations*, February 2006.
20. Stuckey, Nathan C. *Stochastic Estimation and Control of Queues within a Computer Network*. Master's thesis, Graduate School of Engineering, Air Force Institute of Technology (AETC), Wright-Patterson AFB OH, March 2007. AFIT/GE/ENG/07-24.
21. Thompson Eaddie, Marnita. *Dialable Cryptography for Wireless Networks*. Master's thesis, Graduate School of Engineering, Air Force Institute of Technology (AETC), Wright-Patterson AFB OH, March 2008. AFIT/GCO/ENG/08-02.
22. Welch, Greg and Gary Bishop. "An Introduction to the Kalman filter". 2001.
23. Wolski, Rich. *Dynamically Forecasting Network Performance Using the Network Weather Service*. Technical Report TR-CS96-494, University of California, San Diego, 1998.
24. Wolski, Rich, Neil Spring, and Jim Hays. "Network Weather Service: A Distributed Resource Performance Forecasting Service for Metacomputing". *Future Generation Computing Systems*, 1999.

### *Vita*

Cindy Reese is a student at the Air Force Institute of Technology (AFIT), pursuing a Masters Degree in Software Engineering/Electrical Engineering. Captain Reese will graduate from AFIT, March 2009.

Permanent address: 2950 Hobson Way  
Air Force Institute of Technology  
Wright-Patterson AFB, OH 45433

REPORT DOCUMENTATION PAGE				Form Approved OMB No. 074-0188	
<p>The public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of the collection of information, including suggestions for reducing this burden to Department of Defense, Washington Headquarters Services, Directorate for Information Operations and Reports (0704-0188), 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to a penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number.</p> <p><b>PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.</b></p>					
1. REPORT DATE (DD-MM-YYYY) 19-03-2009		2. REPORT TYPE Thesis		3. DATES COVERED (From – To) June 2007-March 2009	
4. TITLE AND SUBTITLE  Decisive Routing and Admission Control According to Quality of Service Constraints				5a. CONTRACT NUMBER	
				5b. GRANT NUMBER	
				5c. PROGRAM ELEMENT NUMBER	
6. AUTHOR(S) Reese, Cindy C., Captain, USAF				5d. PROJECT NUMBER 09-267	
				5e. TASK NUMBER	
				5f. WORK UNIT NUMBER	
7. PERFORMING ORGANIZATION NAMES(S) AND ADDRESS(S) Air Force Institute of Technology Graduate School of Engineering and Management (AFIT/EN) 2950 Hobson Way WPAFB OH 45433-7765 DSN: 785-3636				8. PERFORMING ORGANIZATION REPORT NUMBER  AFIT/GE/ENG/09-36	
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) John D. Matyjas, Program Manager AFRL/RIGE (AFMC) 525 Electronics Parkway, Rome NY 13441 (315) 330-4255 (DSN 587) John.matyjas@rl.af.mil				10. SPONSOR/MONITOR'S ACRONYM(S) AFRL/RIGE	
				11. SPONSOR/MONITOR'S REPORT NUMBER(S)	
12. DISTRIBUTION/AVAILABILITY STATEMENT APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED					
13. SUPPLEMENTARY NOTES					
14. ABSTRACT  This thesis will research, model, and propose options to enhance command and control in for communication networks. My goal is to research the viability of combing past research in the areas of network prediction, context-aware routing, and QoS-based routing optimization in order to create an intelligent routing protocol platform for mobile networks. It will consolidate efforts from an <i>Intelligent Agent Based Framework to Maximize Information Utility</i> by Captain John Pecarina, <i>Dial-able Cryptography for Wireless Networks</i> by Major Marnita Eaddie, and <i>Stochastic Estimation and Control of Queues within a Computer Network</i> by Captain Nathan Stuckey. My thesis will create a platform that is greater than the sum of its individual parts. The platform will take predictions about the health of the network and will take the priority level of a commodity that needs to be routed, and then will this information to intelligently route the commodity in such a way as to optimize the flow of network traffic. Developing this platform will ensure that the forward commander can make intelligent decisions at the right time using the right information and on the right network.					
15. SUBJECT TERMS Communication network, quality of service					
16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT  UU	18. NUMBER OF PAGES  110	19a. NAME OF RESPONSIBLE PERSON Kenneth Hopkinson, PhD (ENG)
REPORT U	ABSTRACT U	c. THIS PAGE U			19b. TELEPHONE NUMBER (Include area code) (937) 255-6565; email: Kenneth.hopkinson@afit.edu

**Standard Form 298 (Rev: 8-98)**

Prescribed by ANSI Std. Z39-18